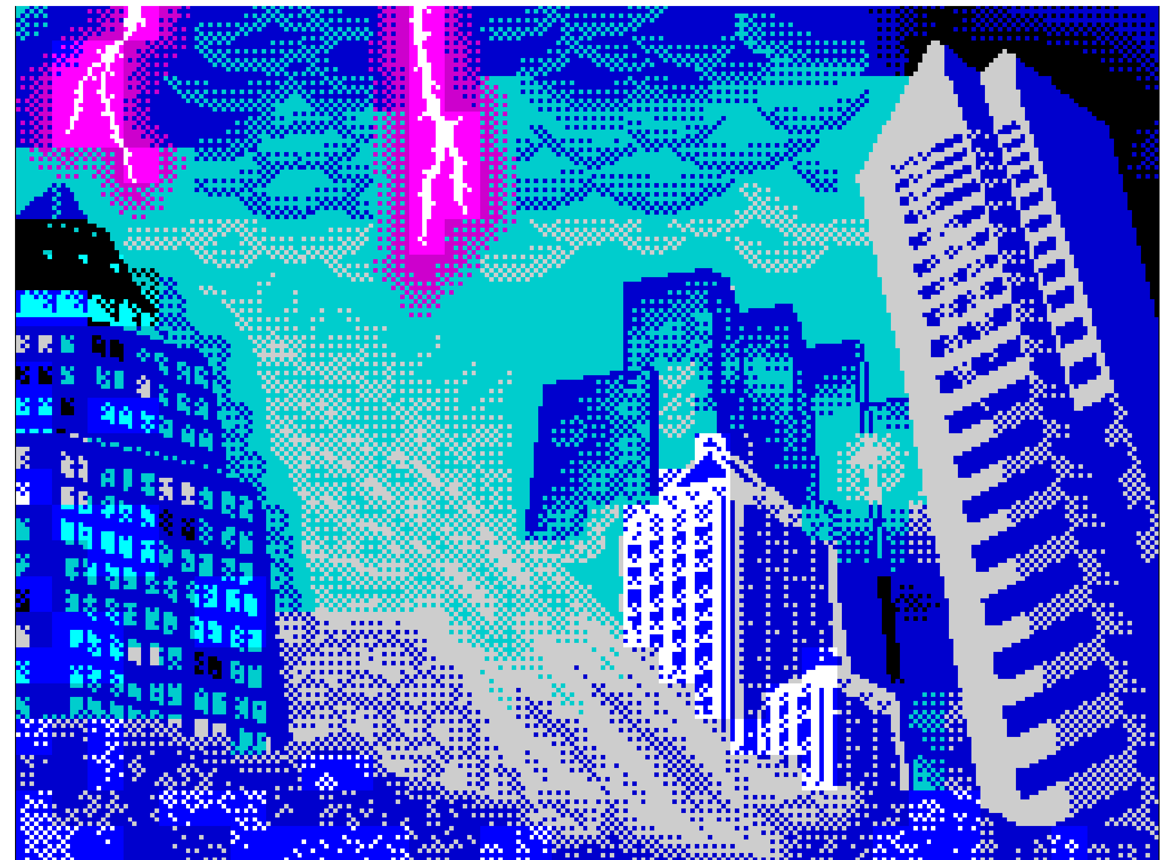




Sensation! SPC übersteht den 21. Dezember 2012

Herausgeber und für den Inhalt verantwortlicher:
Leszek Chmielewski, Prager Straße 92/11/12, 1210 Wien, Österreich
@Mail: retrozx@gmail.com
Klubkonto (Inhaber: Bernhard Lutz):IBAN: DE59 5486 2500 0000 5461 43
SWIFT-Code: GENODE6K, BIC-Code: GENODE61SUW
KTO.: 546143, BLZ: 54862500 (VR Bank Südpfalz, Sitz: Landau)

Ausgabe 231

4 Quartal 2012

Das Vorwort

http://www.womoteam.de/
http://spc.tlienhard.com/

Willkommen zu der Zeitschrift von Usern für User. Wir sind vor allem auf EURE Artikel angewiesen. Ich kann alleine keine (angepeilten) 24-32 Seiten füllen, so gerne ich es auch tun würde. Ehrenwort! Für eingeschickte Artikel gelten folgende Regeln:
Die Artikel müssen sich mit dem Spectrum, ZX81, SAM Coupé, Sprinter 2000 oder nahen Verwandten des Sinclair ZX Spectrum befassen, auch Artikel über passende Hardware und Software sind gerne gesehen.
MAC/PC Software: Nur wenn ausdrücklich direkt im Zusammenhang mit den eingangs erwähnten Computern. Sehr gerne: Crosscompiler, Emulatoren, Game Maker und dergleichen. Auf keinen Fall aber Remakes von Spielen alter Plattformen auf moderner Hardware.

Von Bernhard Lutz kam folgende Nachricht:

ich habe den SPCINDEX mal wieder auf den aktuellen Stand gebracht:

http://www.womoteam.de/ftp/spcindex.htm
bzw.:
http://www.womoteam.de/ftp/spcindex.txt

Nachricht von Yerzmyey:
OK guyz, here is the newest demo for ZX Spectrum 128 -
"Nightmares" by Noice.
http://pouet.net/prod.php?which=59959

Second place on Sundown 2012 party.

Code: Shadow/Noice
Graphics: Frost/Panda Design
Music: Yerzmyey/Hooy-Program

TR-DOS version for diskdrive/Z-Controler users:
http://zxaaa.untergrund.net/demo.php?a=Noice

Enjoy.
Yerz

Des weiteren:

Hi,

http://chipmusic.org/yerzmyey/music/proof-of-concept-2-x-zx-spectrum-chiptune-digi

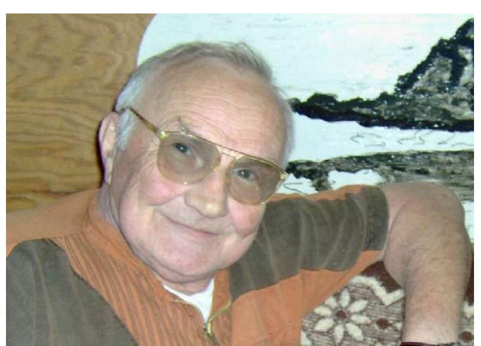
This is really a proof of concept, I made for the idea of playing simultaneously two ZX Spectrum computers, where one is playing chiptune music and the second one is playing digital music (6 channels in total).
I was wondering if it keeps tempo properly - and it appeared it works OK.
I don't know if it sounds very well, but at least sounds kind of interesting, I'd say.

One Spectrum is playing a chiptune part from ZX SoundTracker 1.1.
The second Spectrum is playing a 3-channels 4-bit digi-music from SampleTracker 2.1.

Enjoy.
Byez, Yerz

In tiefster Trauer muss ich bekanntgeben, dass mein Vater am 6.12. im Alter von 69 Jahren verstarb.
Er hatte am 3.12. einen Riss der Aorta und starke innere Blutungen. Der Chirurg sagte, dass es zwei Liter Blut waren.
Die Notoperation überlebte er zwar. Doch am 6.12. versagten aufgrund von

Durchblutungsstörungen die Organe, nicht zuletzt dadurch, dass mein Vater Raucher war und seine Venen nicht im allerbesten Zustand waren.



Durch diese Tatsache bedingt, verzögerte sich das SPC-Clubinfo ganz erheblich.

LCD-Leszek Chmielewski

Termine 2013

19. - 21.04.2013: ZX-TEAM Treffen
D-36145, Hofbieber, (Mahlerts)
Die Hardwareprofis für den ZX81 und ZX-Spectrum mit originellen und innovativen Ideen!

27. - 28.04.2013: Vintage Computerfestival
Mehrzweckhalle des ESV, Baumkirchner Straße 57, D-81673 München
Europa (VCFe 14.0) in München
Hauptthema : Lernen
Mehr unter www.vcfe.org/D/

04.05.2013: Die lange Nacht der Computerspiele
D-04289, Leipzig, Karl Liebknecht Str. 145, Beginn 16 Uhr. Ende um 3 Uhr Sonntagmorgens.
Im Mittelpunkt steht das Ausprobieren alter wie neuer Spiele. Entwickler und Studenten zeigen ihr Schaffen, Sammler ihre Videospiel-Schätze. Die Roboter-Fußballmannschaft der HTWK stellt sich vor. Eine Vitrinen-Ausstellung zeigt historische Spiele und Hardware .... uvm
Weitere Infos unter www.schreibfabrik.de

07. - 08.09.2013: Spectra-Joyce
Wolfhagen
Die Joyce User-AG und der Spectrum Profi Club beschäftigen sich mit CP/M Rechnern (Joyce) und Sinclair Computern

24. - 25. 08.2013: Spectrology
Kulturbund, 06886 Wittenberg, Lutherstr 43 a
Schwerpunkt sind der ZX-Spectrum und ZX81, aber auch andere 8-Bitter Freunde haben dieses Treffen in der Lutherstadt für sich entdeckt!

noch offen: Xcentrix
Seeshaupt, Oberbayern
Das 16. XzentriX Treffen wendet sich an Computerfreunde, die an ein Leben nach der Standardsoftware glauben.

Foreword-English

Welcome to the magazine by users for users. We are primarily dependent YOUR article. I alone can not fill (targeted) 24-32 pages, even I would like to do it. Word of honor! To be sent article, the following rules:

The articles have to deal with the Spectrum, ZX81, SAM Coupe, Sprinter 2000, or close relatives of the Sinclair ZX Spectrum, including articles on appropriate hardware and software are welcome.
MAC / PC software: Only if expressly directly in connection with the above-mentioned computers. I would be very happy about: cross compiler, emulators, Game Maker, and the like. In no case, however, remakes of old games on New Plattforms.

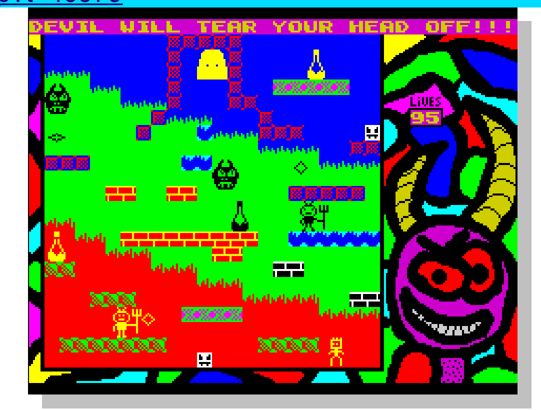
In the deepest sadness I must announce that my father died on 6.12 at the age of 69 years.
He had at 3.12. a tear of the aorta and massive internal bleeding The surgeon said that there were two liters of blood. Although he survived the emergency surgery. But on 6.12 due to circulatory problems, the organs have failed, not least because my father was smoking and his veins were not in the best condition.
Because of this fact due to the SPC Club info was significantly delayed.

LCD Leszek Chmielewski

Neuigkeiten für unseren „Alten“

Schweinereien

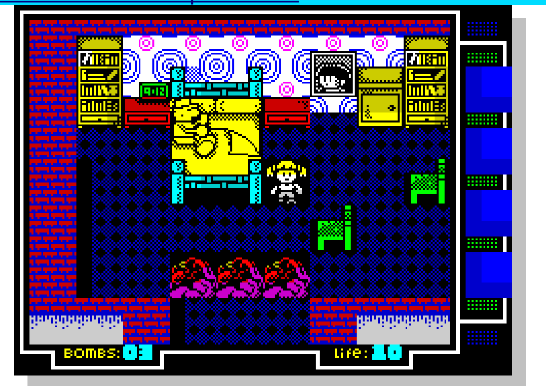
http://www.worldofspectrum.org/forums/showthread.php?t=40979



Hooy-Program haben ein neues AGD 3 Platformspiel releast „Zbylut Owrzodzien w kamiennym, kurwa, zajebanym czarnym kregu“, auf englisch „Crapbert Buttslut in the muthafuckin' damn stone-circle of the devil“. Das Spiel ist höllisch schwer, macht aber Spaß und der AY Soundtrack ist ein echter Ohrwurm.

Maritrini prequel

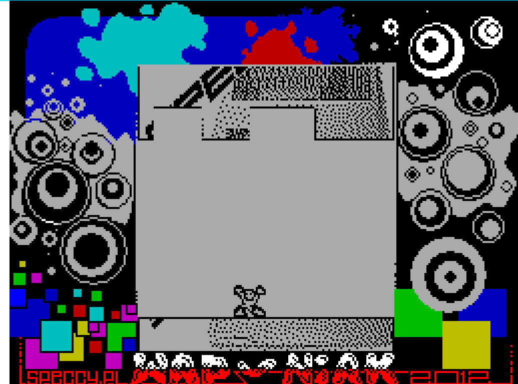
http://www.mojontwins.com/juegos_mojonos/maritrini-freelance-monster-precuela/



Mojon Twins haben es auch irgendwie mit den langen Titeln. „Maritrini, Freelance Monster Slayer en: las increibles vieisitudes de despertarse resacosa con fred en la cama y tener que llegar mas o menos puntual a la prueba de >>monstruos vigoroses de pechos lustrosos<< featuring los fratelli“. Hinter diesem Titel verbrigt sich ein grafisch sehr schönes (teilweise geklaute Grafiken aus Fred, Batman und Psycho Pig UXB) in ZXBC geschriebenes Spiel für den Spectrum 48. Es ist die Demonstration eines neuen Framewowks für ZXBC, das die Spieleherstellung erleichtern soll. Ich finde es jedenfalls ganz super, bis auf den viel zu langen Titel.
Der Source Code ist übrigens auch verfügbar.
Der Titel, frei ins Englische übersetzt, bedeutet übrigens: „Maritrini, Freelance Monster Slayer, in «The incredible misadventures of waking up with a hangover and Fred in your bed and having to get (more or less in time) to an audition for "Vigorous Monsters with Shiny Manly Chests"», featuring The Fratelli Family„.

Gehe zum Loch

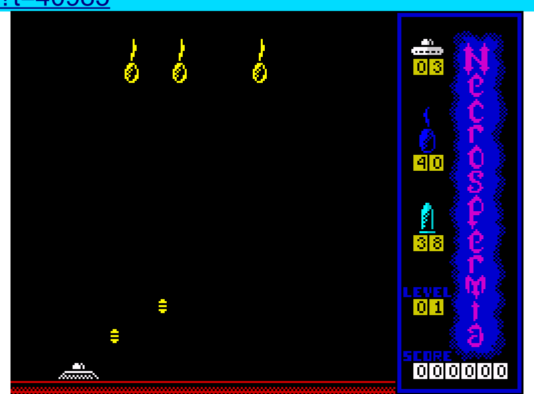
http://www.worldofspectrum.org/infoseekid.cgi?id=0027945



Tygrys, Cat Man und Voyager aus Polen haben „Dziurak – Go to Hole!“, ein Pre-Release der Wapniak Party veröffentlicht. Nett präsentiert, geht es im Spiel darum die Spielfigur in einer ausreichend großen Lücke zu platzieren bevor die Decke runterkommt.
Manchmal sind die Löcher in der Höhe sehr knapp bemessen.

Necrospermia

http://www.worldofspectrum.org/forums/showthread.php?t=40985

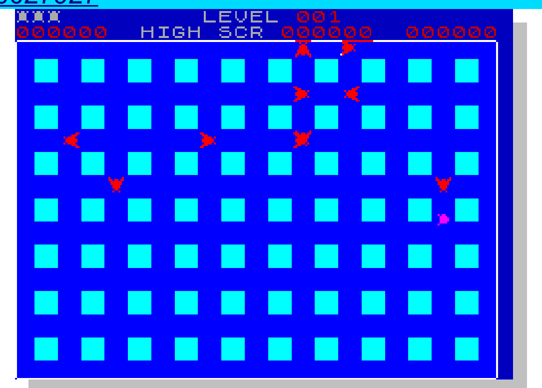


Ralfs neuestes Werk ist sehr simpel. Man schießt auf Spermien die von oben nach unten fallen. Leider hat das Spiel ein großes Problem: Die Tastatur-Steuerung funktioniert nicht. Hoffentlich wird es bald gefixt.



Mole Rat!

http://www.worldofspectrum.org/infoseekid.cgi?id=0027927

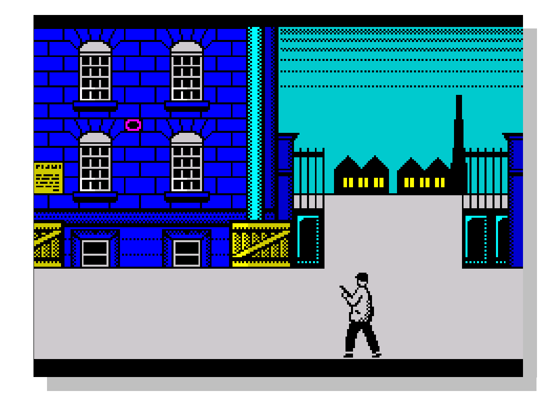


Stonechat Productions (Dave Hughes) hat ein nettes Labirynthspiel für den Spectrum entwickelt welches sich hinter schlichter Grafik verbirgt.

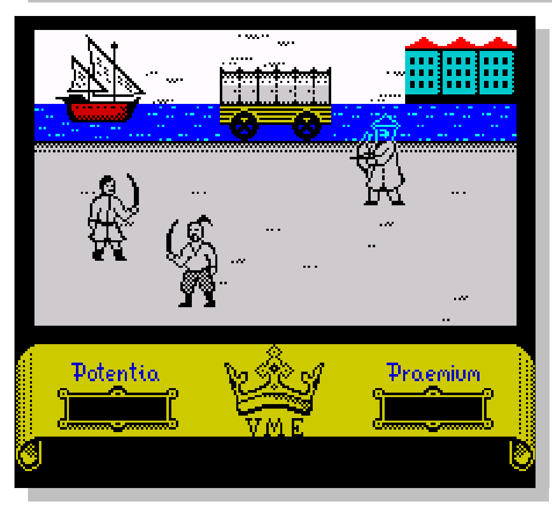
Ralfs Projekte

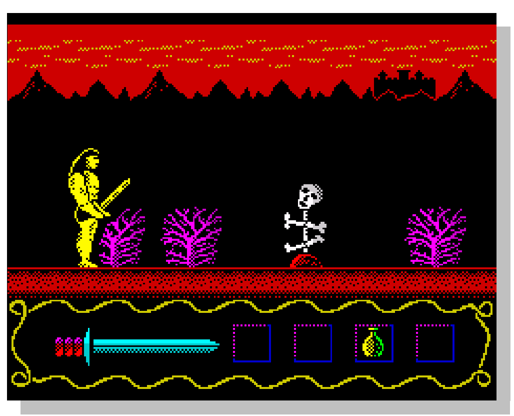
http://www.worldofspectrum.org/forums/showpost.php?p=656399&postcount=4

Ralf hat die Screenshots einiger seiner Projekte veröffentlicht. Alles ohne Garantie, dass die Programme je erscheinen werden. Auf jeden Fall sehr interessante Einblicke.









Stormfinch

http://www.worldofspectrum.org/forums/showpost.php?p=656458&postcount=9



Ein Scrollendes Shoot'em'up welches mit 50 bildern pro Sekunde butterweich scrollen soll, hat derzeit R-Tape in der Mache. Leider ist nur eine einzige Art von Tile bei der Implementierung erlaubt. Ich bin schon auf das Endergebnis gespannt. Hoffentlich wird das Spiel was Reißen, sonst heißt es wieder „Außer Spesen nichts gewesen".

BiNMaNs Werke

http://www.worldofspectrum.org/forums/showpost.php?p=656511&postcount=15

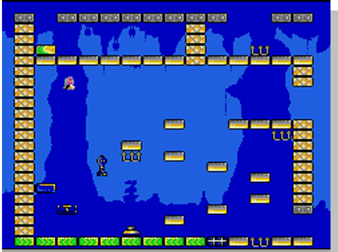
Die Projekte von BiNMaN die er auf WOS vorgestellt hat, sehen sehr interessant aus:



Shooter und ein Auto-Spiel in der Machart von Spy Hunter (mit Option für ULA+) sehen hervorragend aus. Es muss aber noch viel daran gearbeitet werden. Und ja, eines der Autos auf den Bildern ist ein VW Käfer!

Spuds am Spectrum & SAM Coupé

http://www.worldofspectrum.org/forums/showthread.php?t=41619



Spud hat sein Spiel für SAM Coupé und Spectrum fertig geschrieben – Dave Infuriators. Das Besondere an diesem Plattformer ist, dass er mit nur einer Taste gesteuert werden kann. Die Figur kehrt automatisch an den Wänden um, man kann nur auf Tastendruck springen. Wer will, kann mit der zweiten Taste die Figur vorzeitig umkehren lassen. Verschiedene Plattformen haben unterschiedliche Auswirkungen. Das sieht nach einem sehr herausfordernden Spiel aus.
Beide Versionen sind gleichzeitig erschienen, wie in guten alten Zeiten.

Flucht aus Cleveland

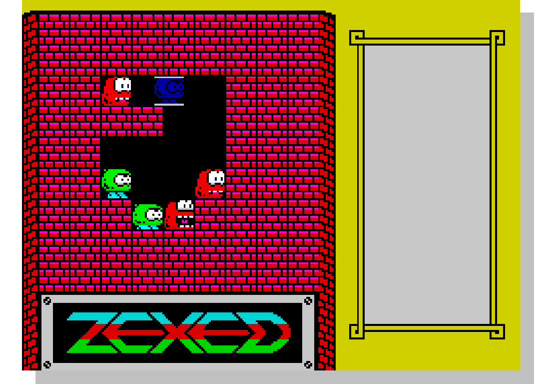
http://www.worldofspectrum.org/forums/showpost.php?p=656561&postcount=26



Judas of EZT schreibt mittels AGD an dem neuen Plattformer: „Escape from Cleveland", basierend offenbar auf „Escape from L.A.". Leider ist der Autor ein fauler Sack. Deswegen kann es bis zum Release noch etwas dauern.

Hires Color Puzzler

http://www.worldofspectrum.org/forums/showpost.php?p=656808&postcount=40



„ZEXED“ ist ein Puzzler von Einar Saukas (dem Author der Bifrost Engine). Es war monatelang still um das Projekt, doch nun sieht es so aus als ob die Arbeit daran wieder fortgesetzt wird. Go, Einar, Go!

Rotkäppchens brutaler Bruder

http://www.worldofspectrum.org/forums/showpost.php?p=656930&postcount=49

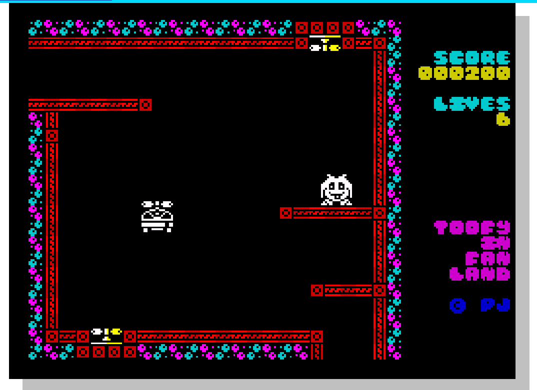
Daveysludge arbeitet an einem sehr farbenfrohen Plattformspiel mit dem vertraut klingenden Titel „Red Beret II – The Death Squad“.
Ein politisches Statement zu diesem Titel sollte ich mir besser sparen. Auf dem Titelbild sieht man einen Weissen der einen Chinesen zusammenschlägt.



Unbeeinflußt davon glaube ich, dass das Spiel Potential hat. Die Grafiken sehen sehr detailliert aus, und die Farbwahl passt, bis auf das etwas unglückliche Titelbild.

Toofy ist nicht doofy

http://www.worldofspectrum.org/forums/showthread.php?t=40743



„Toofy in fan land“, ein neues AGD-Spiel von Paul-J ist nun fertig, und bietet ein außergewöhnliches Spielerlebnis mit Gravitationsumkehrung (eigentlich drücken einen nur die Ventilatoren an die Decke, komischerweise dauerhaft).
Übrigens arbeitet Jonathan gerade mit Kiwi an einem PC-basierenden AGD Crossplattform-System.

Krakos

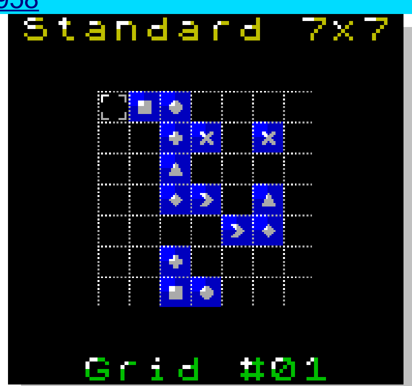
http://kabutofactory.netne.net/index.php/juegos/10-los-buenos/10-arcos-todas-las-versiones



Kabuto Factory unter Baron Ashler hat das Krakout-ähnliche Spiel A.R.C.O.S. endlich fertiggestellt, wobei sogar halbwegs flüssig scrollendes „Paralax-Background“ hinzugefügt wurde (die Routine stammt von Ralf). Das Spiel sieht aus als ob Krakout es mit Space Invaders getrieben hätte. Für vermutlich compiliertes BASIC gar nicht mal so übel. Eine ZX81 Version gibt es auch.

Drag King

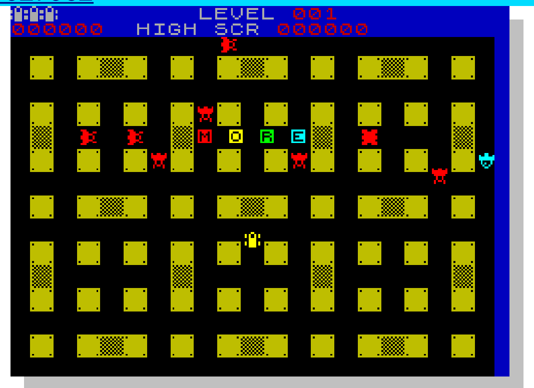
http://www.worldofspectrum.org/infoseekid.cgi?id=0027958



Tom Dalby, der Autor der Hits wie „Flynn's Adverntura in Bombland“ hat mit Drag'nJoin 4K ein neues Mini-Spiel in 4 KB programmiert. Das Ziel ist es, alle gleichen Symbole miteinander zu verbinden, ohne dass sich die Linien kreuzen. Knifflig aber sehr gut! Vor allem gibt es mehrere Schwierigkeitsstufen zur Auswahl.

Shuttlebug

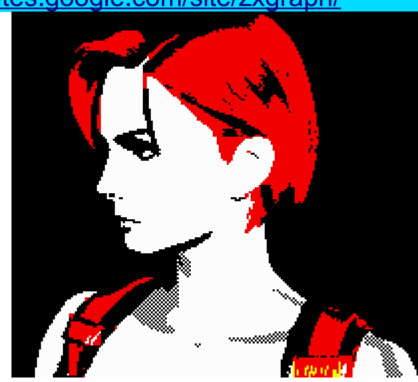
http://www.worldofspectrum.org/infoseekid.cgi?id=0027962



R-Tapes neuestes Spiel heißt „Shuttlebug“ und ist sehr interessant. Man fährt im Labirynth, schießt Gegner ab, achtet dass man nicht von Bombenexplosionen erwischt wird... Sehr gut programmiert, kann ich nur empfehlen! Die Grafiken sind sehr flüssig.
Aber keine Sorge! Das Spiel macht selbstverständlich auch viel Spaß.

ZX Grafik Bibliothek

https://sites.google.com/site/zxgraph/



Einar Saukas hat sich dazu entschlossen eine Website aufzumachen wo Artisten ihre Werke wie Spritesheets, Tilesheets, Grafiken und ähnliche, Programmierern für ihre Spiele zur Verfügung stellen können. Eine hervorragende Idee, wie ich finde. Bitte unterstützen!

Euphorisch!

http://zx.pk.ru/showthread.php?t=20370



Euphoria 2D ist wie angekündigt fertiggestellt und das was Civilisation am nächsten kommt. Von den anfänglich bemängelten Schwächen ist nichts mehr geblieben, und es spielt sich wirklich sehr gut. Sourcecode wurde auch veröffentlicht.
Das Spiel gibt es als TAP sowohl auf Russisch wie auch auf Englisch, und es funktioniert auf dem Spectrum 48K. Als Fan von Strategiespielen ist es meine Pflicht dieses zu empfehlen. Ihr werdet es nicht bereuen.

Der Angriff geht weiter

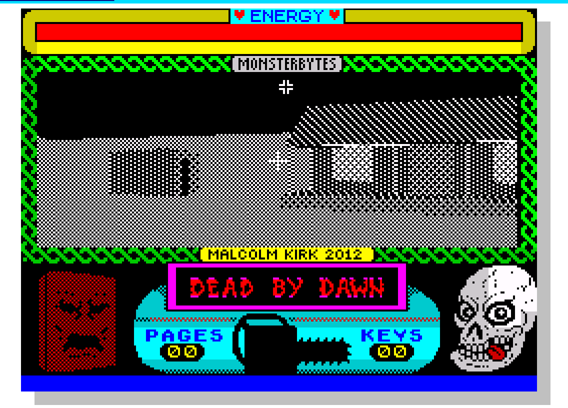
http://www.worldofspectrum.org/infoseekid.cgi?id=0026121



Ich habe mein Chessboard Attack ein wenig aufpoliert und die Version 1.1 zur Verfügung gestellt. Die neuen Features konnte ich innerhalb nur eines Tages einbauen: Schnellere Routinen, mehr Grafiken, neuer Option-Bildschirm, abschaltbare Musik und Beeper-Effekte, definierbare Tastenbelegung bzw. Kempston Joystick. Das ging so schnell weil die meisten dieser Routinen aus „Yumiko in the haunted Mansion“ transferiert wurden. Außerdem habe ich die Arbeiten an „Chessboard Attack II“ angekündigt, welches eine andere Idee zugrunde liegen hat.

„Ein Tod am Morgen bringt...“

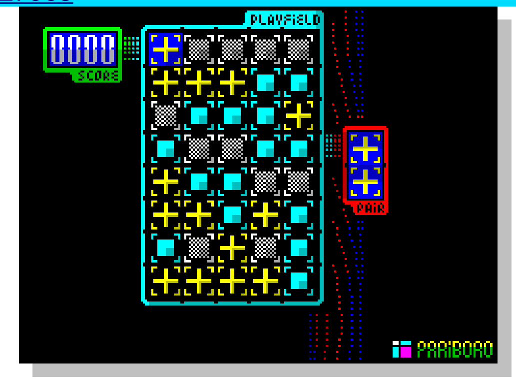
http://www.worldofspectrum.org/infoseekid.cgi?id=0027972



Malcolm Kirks „Dead by dawn“ ist nun endlich fertiggestellt und präsentiert sich als ein geniales Horror Adventure mit der „Freescape“ Engine.
Das muss man einfach spielen!
Malcolms Blog findet man übrigens hier:
http://malcolmkirk.blogspot.co.at/

Pärchen verkuppeln

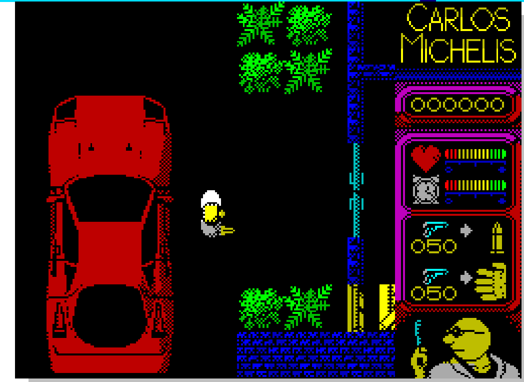
http://www.worldofspectrum.org/infoseekid.cgi?id=0027963



„Pariboro“ ist das neueste Spiel vom Zero Team aus der Slowakei.
In diesem Puzzler muß der Spieler umgekehrt im Vergleich zu Tetris, Reihen freiräumen indem man vorgegebene Paare vertikal oder horizontal sucht. Ist kein Paar im Set zu finden, heißt es Game over.
Ein sehr gut gemachtes Spiel welches noch obendrein als Sourcecode angeboten wird.

Official Agent Man!!!

http://www.worldxxisoft.com/En/CM/cm_en.htm



„Carlos Michelis (4)“ heißt der neueste Superhit von World XXI Soft Inc. In diesem 300KB schweren farbenfrohen Action-Spektakel muss man als ein

Agent die Tochter des Präsidenten befreien (die Sequenz wo Bruno ihr die Hand verstümmelt ist nichts für schwache Nerven). Das Spiel ist so eine Art Mischung aus „Into Eagles Nest“, und 2D Variante von „Doom“.
Es gibt drei nachladende Levels, riesige Explosionen, High Tech Mechas, und vieles mehr. Insgesamt ein tolles Spiel. Die digitale Kopie kostet zwar 5 US-Dollar, aber nachdem diese Währung ohnehin nicht das Papier wert ist auf dem es gedruckt wurde, ist es gut angelegtes Geld. Den Kauf bereue ich jedenfalls nicht.
Es gibt natürlich ein kostenloses Demo für alle die sich von den Spielqualitäten überzeugen wollen.
Es wurde übrigens mit SDV2 geschrieben, einem Entwicklungs-ROM das in Entwicklung ist.

ASCII-Kriege

http://www.bobs-stuff.co.uk/lumascii.html

Bob hat sein LumASCII fertig und verkauft dieses schöne Spiel für eine freiwillige Spende ab 3 Pfund aufwärts.
Es handelt sich hierbei um ein Spiel das R-Type ähnelt, jedoch statt Bitmap-Grafiken, die ASCII Zeichen des Spectrum verwendet. Es funktioniert trotzdem sehr gut und hat einen eigenen Charme, dem man sich kaum entziehen kann. Aufgrund der Tatsache dass keine Bitmaps den Speicher belegen, lassen sich mehr Code & Levels unterbringen.

Galaktisches Nachschlagewerk

http://www.retrofusion.me.uk/index.php?page=shop.product_details&flypage=vmj_naru.tpl&product_id=23&category_id=6&option=com_virtuemart&Itemid=25&vmcchk=1&Itemid=25



„Encyclopaedia Galactica“ ist nun fertiggestellt. Jonathan Cauldwells neuestes Spiel in dem man Aliens katalogisieren muss, steht für 2 Pfund auf der Retrofusion Seite zum Download bereit. Aber Achtung!!! Mein Download Link führte mich zu einem Z80 Snapshot. Als ich Wirbel gemacht habe, schickte mir Chris wortlos eine TZX Version zu, nur dass diese defekt war und die 128K-Zusatzdaten nicht lud (quasi 48K Version). Erst eine Beschwerde bei Jonathan Cauldwell mit Kaufnachweis führte dazu, dass ich eine korrekt funktionierende Version als TZX bekam.
Also wenn es Euch auch passiert, wendet Euch an Jonathan. Er wusste gar nicht, dass Chris sein Werk als Snapshot verkauft.
Das Spiel selbst ist das Geld auf jeden Fall wert. Ich hoffe nur dass es nicht in Mode kommt, dass Snapshots verkauft werden.

Z88dk

http://www.z88dk.org/forum/viewtopic.php?id=5176

Das ANSI C Entwicklungssystem Z88dk bekam mit der Version 1.10 ein neues major Release mit vielen Verbesserungen.

Boriels ZX BASIC Compiler

http://www.boriel.com/wiki/en/index.php/ZX_BASIC:Archive#Latest_Development_Version

Boriels ZXBC wurde auf Version 1.3.0s938 upgedated. Viele Bugs wurden beseitigt und es ist schneller!

Der Hobbit und andere RPGs

http://foro.speccy.org/viewtopic.php?f=6&t=2965&p=31171#p31171



Der von mir erwähnte Bytemaniacos BASIC RPG Wettbewerb trägt erste Blüten. „El Hobbit“ und ein paar weitere Spiele sind am Entstehen.



Ich werde über die Ergebnisse auf jeden Fall berichten.

Pixlige Welten

http://www.worldofspectrum.org/forums/showthread.php?t=41608

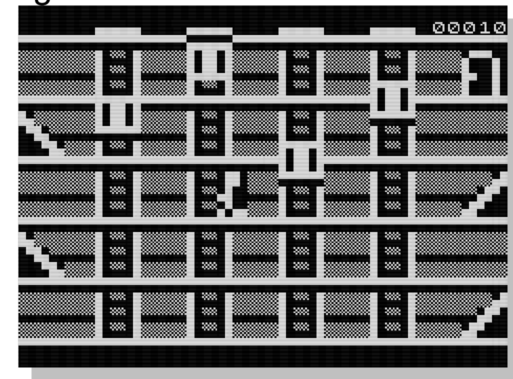


Pixel World ist die Umsetzung eines Flash-Spielchens vom PC am Spectrum, welche Goblinish schreibt. Jedoch nachdem der User Sicherheitslücken auf zx.pk.ru aufgezeigt hat, wurde er dort gesperrt und so ist zu befürchten, dass dieses geniale Spiel niemals an die Öffentlichkeit kommt.

Nach unten bitte!

http://www.revival-studios.com/?page=135
http://www.revival-studios.com/?page=133

Down! Ist ein neues Spiel für Sinclair ZX81 mit 16K von Revival Studios. Ich kann nicht viel dazu sagen außer dass der Download € 3,99 kostet, während die Kassette mit € 7,99 natürlich etwas teurer ist, aber ein schönes Cover hat, welches denen aus damaliger Zeit im nichts nachsteht. Zu dem Spiel selbst fragt mich nicht! Ich habe es nicht gekauft und auch nicht gespielt.
Dieselben Preise gelten auch für das zweite Spiel: „Stair Runner“, auch das habe ich mir nicht geleistet, schaut aber sehr gut aus.



AGD Glory

http://www.worldofspectrum.org/forums/showthread.php?t=39085&page=3



SimonLCFC hat die Vorschau auf sein AGD-Projekt online gestellt. Noch ist es unbenannt. Die Grafiken sind aber auf jeden Fall hochqualitativ.

Verschollen im Spectrum

http://www.worldofspectrum.org/infoseekid.cgi?id=0027974



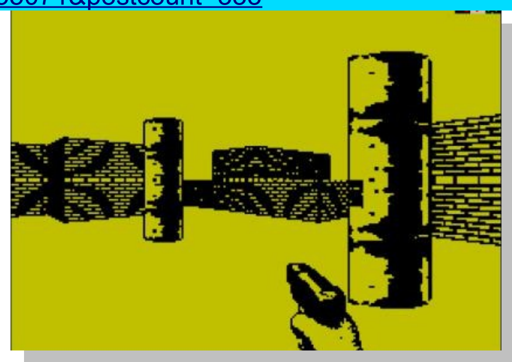
Allesandro Grussu hat sein erstes Spectrum Spiel fertiggestellt. Es schaut wie Manic Miner aus, und spielt sich fast genauso, ist aber um einiges Farbenfroher als das Original.

„Lost in my Spectrum“ wurde mit AGD geschrieben und läuft am Spectrum 48 und 128. Zwei Sprachversionen sind verfügbar: Englisch und Italienisch. Gesteuert wird es mit nur drei Tasten.

Einige der Screens sind Tribute an bekannte Spectrum-Spiele wie Arkanoid, oder Heartbroken.

ZXOOM als TAP

http://zx.pk.ru/showpost.php?p=555071&postcount=333



Von dem beinahe-First Person Shooter Zxoom hat Andrew771 eine TAP Version erstellt, so dass auch Benutzer westlicher Geräte das tolle Spiel nun spielen können.

Canabalt Lite

http://cgc.zx.gen.tr/index.php?game=0710143527

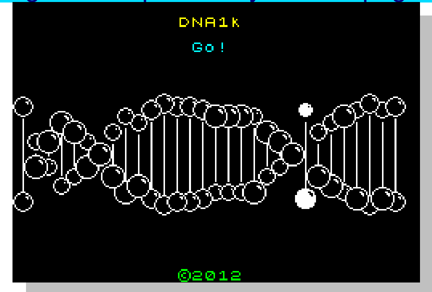


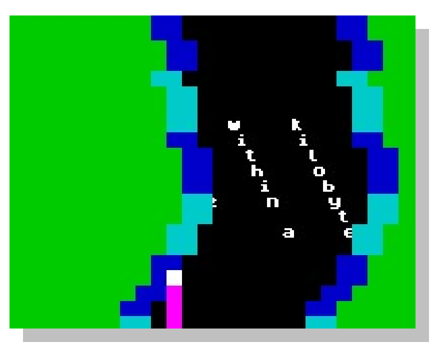
Normalerweise ignoriere ich Spiele, die bei Comp.Sys.Sinclair Crap game Compo eingereicht wurden, doch PODEWRRR! Von Na_th_an ist eine Ausnahme. Es ist sogar recht spielbar.

Die sympholorische Sprache ist das Einzige was mich an dem Spiel stört.

1 Kb Spiele

http://minigamecompo.weebly.com/1k-page1.html





„DNA“ von Tom Dalby und „Slither Highway“ von Andy J. Sind zwei neue ZX Spectrum Spiele in der 1 KB Kategogie hochgeladen worden. Beide sind sehr gut, obwohl das Prinzip von DNA interessanter ist.

SAM Pi

http://programandala.net/en.picture.2012.11.25.raspberry_pi

Sim Coupé, der SAM Coupé Emulator wurde erfolgreich für Raspberry Pi kompiliert.

Projekte von Climacus

http://programbytes48k.wordpress.com/2012/10/09/adelanto-de-los-proximos-proyectos-de-climacus/





Climacus arbeitet an einer Spielengine die mit SCUMM Ähnlichkeiten aufweist und Spiele im Stil von „Secret of the Monkey Island“ ermöglichen soll.
Das zweite Projekt sieht aus wie isometrischer Tapper.

SAM Disk 3.4

http://simonowen.com/samdisk/

Simon Owen hat SAM Disk 3.4 freigegeben. Mit dem Tool lassen sich SAM und +D Disketten lesen und schreiben solange man nicht zu den Pechvögeln zählt, deren Floppys über USB angeschlossen sind.

Hitler-Simulation

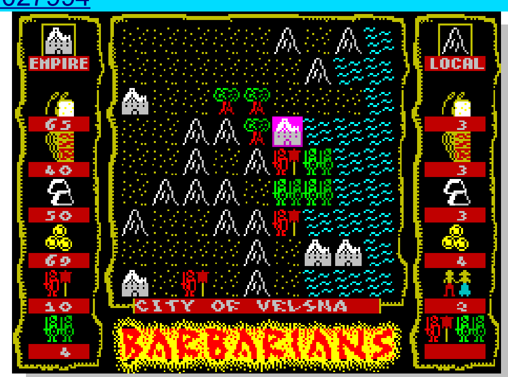
http://www.worldofspectrum.org/infoseekid.cgi?id=0027975



Keine Ahnung ob alle den Adolf Hitler kennen. Er ist ein bekannter Verbrecher, der bis 1945 viele Menschen ermordet hat.
„The world war simulator Part 1“ trägt jedenfalls den Titel „Hitler“, so dass einige Russen geglaubt haben, das ist ein Spiel aus Deutschland. Ich habe den Irrtum aufgeklärt. Dieses „Retrobytes“ Spiel stammt aus Spanien und ist ein Multiple Choice Grafik-Adventure mit Action-Sequencen und vermutlich auch Erotik.
Das Spiel wurde in ZXBC geschrieben und die Source Codes liegen vor, müssten aber etwas optimiert werden...

Barbarenangriff

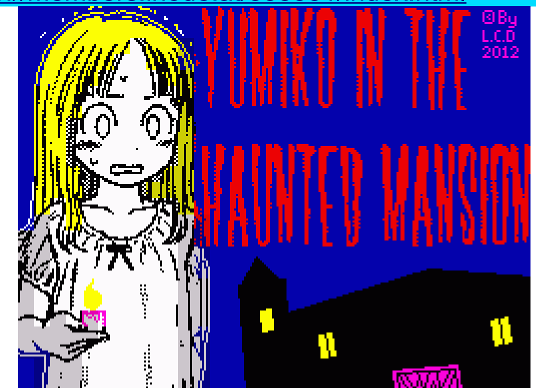
http://www.worldofspectrum.org/infoseekid.cgi?id=0027994



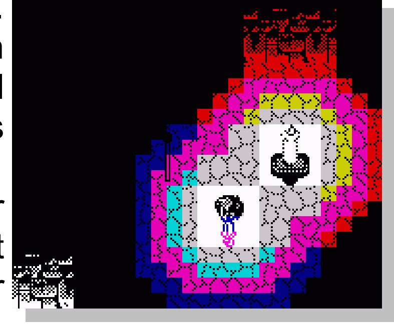
Snigfarp hat gerade ein Spiel, welches er für den Psion 2010 geschrieben hat, für Spectrum 16K (!) portiert.
Im Spiel „Barbarians“ muss man Ressourcen zur Abwehr eines Barbarenangriffs sammeln. Es ist ein Strategiespiel ähnlich meiner „Earthraid“-Umsetzung.

Yumikos erster Auftritt

http://members.inode.at/838331/index.html



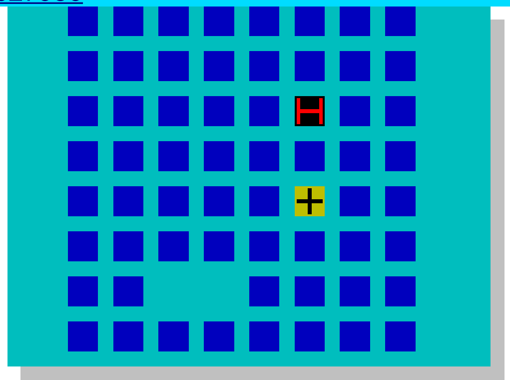
„Yumiko in the haunted Mansion“ ist nun endlich fertig. Es ist mein zehntes Spiel und etwas Besonderes. Die Spieler sind begeistert von der Atmosphäre.



Worum es geht? Nachdem Yumikos Eltern eine Autopanne hatten, sind sie gezwungen in einem Herrenhaus zu übernachten. Als sie die Lichter ausmachen, verschwinden sie plötzlich. Da Yumiko ihr Licht nicht ausgemacht hat, ist sie noch immer da und muss in die Tiefen des Hauses vordringen um in 32 Levels alle Kerzen anzuzünden. Keine leichte Aufgabe, da immer wieder Geister auftauchen und die Kerzen ausblasen.
Das Spiel wurde in ZXBC geschrieben und ist technisch wirklich Top mit den Lichteffekten. Vermutlich momentan das maximum Erreichbare mit ZXBC.
Momentan ist die „Standard-Edition“ verfügbar, aber es wird eine „Collector's Edition“ zu kaufen geben.
Ich plane bereits Nachfolger: „Yumiko in the realm of Slenderman“ und „Yumiko vs: Kreddy Frueger“.

512 Byte Memory

http://www.worldofspectrum.org/infoseekid.cgi?id=0027986



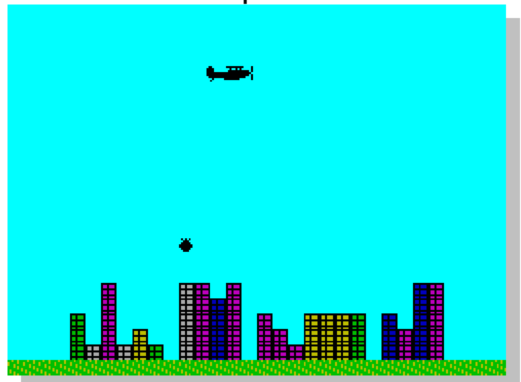
Ralfs Beitrag zu Next Castle 512 Byte game compo ist ein Memory-Spiel. Wegen der Symbole ist es etwas schwer, aber sehr gut programmiert.

Ultra Bomber

http://www.worldofspectrum.org/infoseekid.cgi?id=0027980

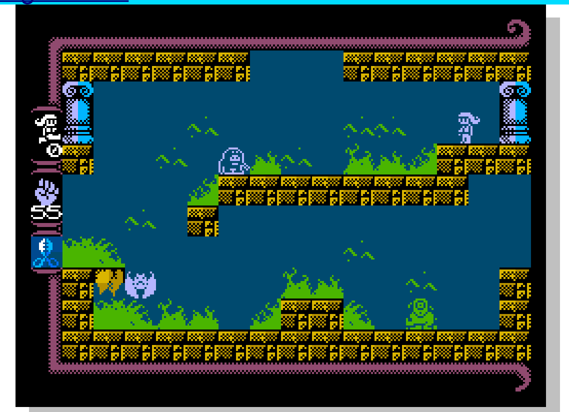
Den zweiten Platz auf der Next Castle Party 2012 512 Byte Demo Compo belegte „Ultra Bomber“ von Dmitry Egorov. Es ist ein „Blitz“-Klon bei dem

man eine Stadt niederbomben muss um zu landen. Der Source Code ist verfügbar, daher ist es nicht so problematisch dass das Spiel nur als ein Snapshot vorliegt, da man sich eine TAP Version selbst kompilieren kann.



Cheril die Göttin

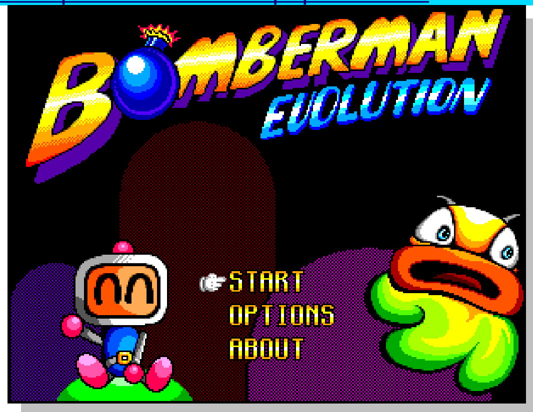
http://www.mojontwins.com/juegos_mojonos/cheril-the-goddess/



Langsam geht die Emanzipation zu weit. Jetzt wollen auch schon Frauen göttliche Kräfte haben.
In „Cheril the Goddess“ (mit ULA+ Support) hat Cheril Fliegen gelernt und kann nun konzentrierte Energie abfeuern. Jedoch kosten diese Fähigkeiten viel Lebenskraft. Also muss sie eine göttliche Substanz abfangen und einnehmen, um diesen Nachteil nicht mehr zu haben, um ihre Nemesis „Serious Monkey“ zu besiegen.
Fliegen will gelernt sein, kurze Stöße müssen reichen um nicht zu viel Energie zu verbrauchen. Tolles Spiel!

Bombermans Rückkehr

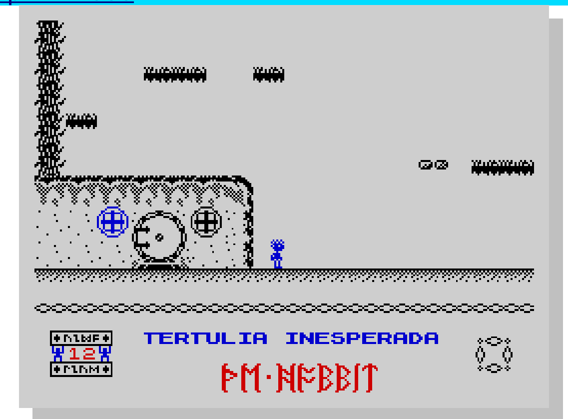
http://zx.pk.ru/showthread.php?t=20412



ZX Evolution (PentEvo) Besitzer dürfen sich über eine hervorragende Umsetzung von Bomberman freuen. Das Spiel läuft mit 50 fps und bietet mit 16 Level viel Abwechselung, u.a. Boss-Kämpfe. Mit „Unreal Speccy“ Emulator kann man das Spiel spielen auch ohne PentEvo zu besitzen.

Der Hobbit

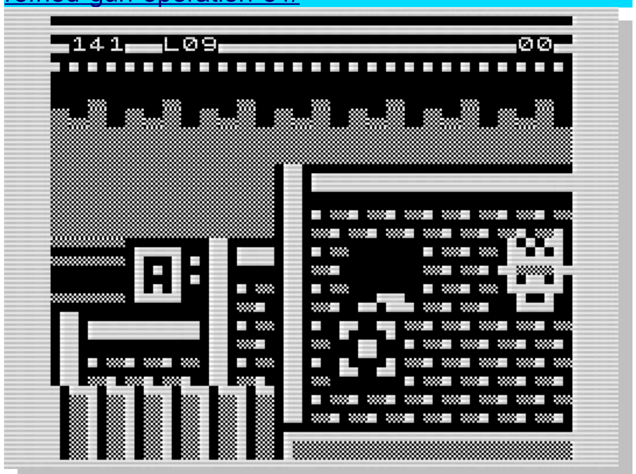
http://www.worldofspectrum.org/forums/showthread.php?t=41830



„El Hobbit“ ist ein neues Spiel von JBGV für Bytemaniacos „Hobbit“ Compo welches neben dem BASIC-RPG Intro parallel läuft, und ist in ZXBC geschrieben worden. Das merkt man ihm nicht an, denn die Sprites und andere Grafiken bewegen sich flüssig und das Spiel erinnert durchaus an „Jet Set Willy“. Leider wurde der Source Code nicht veröffentlicht.

Operation Pistole

http://www.mojontwins.com/juegos_mojonos/super-refried-gun-operation-81/

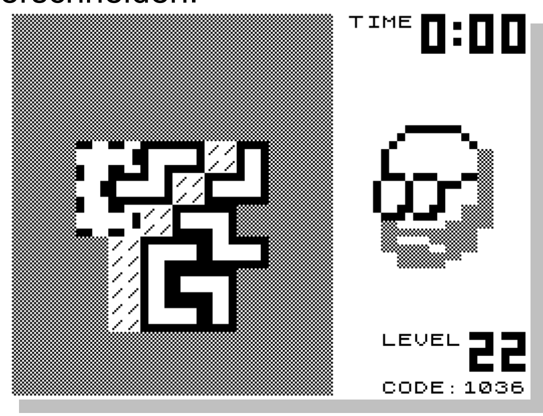


Auch der betagte ZX81 bekommt dank Z88dk immer wieder neues Futter. „Super refried gun Operation '81“ von Mojon Twins ist auch so ein Beispiel. Leider sind die Sources verloren gegangen, weswegen das Spiel unvollendet bleiben wird.

Noir Shapes

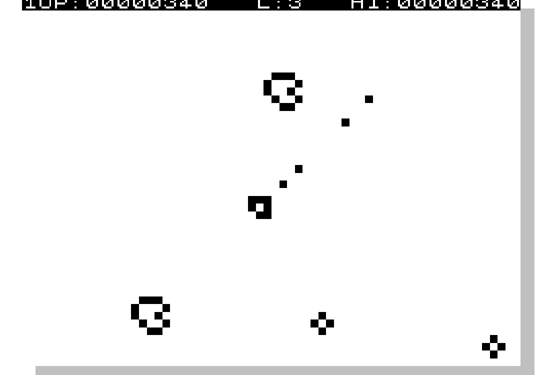
http://www.bobs-stuff.co.uk/noirshapes.html

Bob hat, von allen unbemerkt, einen weiteren Puzzler für den ZX81 programmiert: „Noir Shapes“. Das Spiel basiert auf „Cool Shapes“ für die Microsoft xBox-Konsole. Das Spiel kann man gratis herunterladen.
Generell geht es darum, Figuren so zu drehen, dass sie sich nicht überschneiden.



Einschlag!

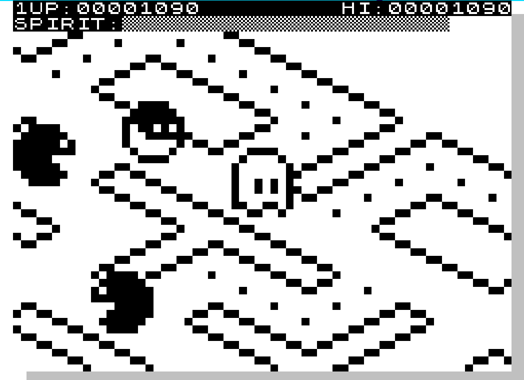
http://www.bobs-stuff.co.uk/impact.html



Mit Impact! Hat Bob wieder still und heimlich ein kostenloses ZX81 Spiel programmiert. Dieses mal bediente er sich dem großartigen „Asteroids“ als Vorbild.
Ehre wem Ehre gebührt! Programmieren kann Bob auf jeden Fall.

Ein einsamer Geist.

http://www.bobs-stuff.co.uk/onelittleghost.html



Der dritte im Bunde der ZX81 Neuerscheinungen von Bob ist „One little Ghost“, ein Pac-Man Klon mit Rollenumkehr. Flüssige isometrische scrollende Grafik hat man bisher so nicht gekannt. Das ist ein Spiel für das es sich lohnt einen ZX81 mit SD-Karte auszustatten. Da sieht man wieder dass keine Hires Grafik oder Farben nötig sind, um Spiele zu spielen, die begeistern können.
Auch dieses Spiel gibt es kostenlos.

Horror in BASIC

http://www.worldofspectrum.org/forums/showthread.php?t=41670
http://www.worldofspectrum.org/infoseekpub.cgi?regexp=^Steve+Westwood$&loadpics=1





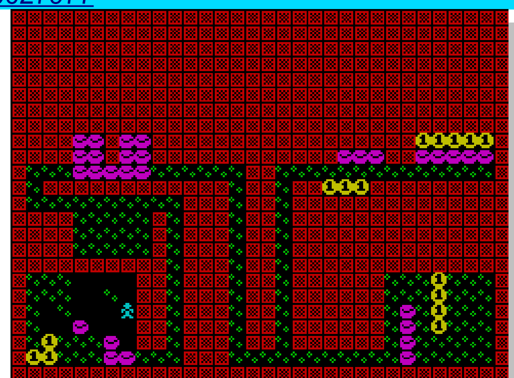




Steve Westwood hat seine Werke wiedergefunden und freigegeben. Es handelt sich um vier Horrorspiele die in BASIC geschrieben wurden.
Ich habe schon schlimmere Spiele gesehen, aber der langsame Bildaufbau muss nicht sein, selbst bei BASIC.

Boulder clash

http://www.worldofspectrum.org/infoseekid.cgi?id=0027977



Next Castle Party brachte uns ein paar nette Spiele. Eines davon ist „Boulder-Dash 512b“, und wie der Name es schon sagt, handelt es sich hierbei um ein Boulder Dash Spiel in nur 512 Bytes. Als Autor ist krt17.

World XXI schlägt wieder zu

http://www.worldofspectrum.org/infoseekid.cgi?id=0027984
http://www.youtube.com/watch?v=pYiyUDqrLhI
http://www.worldofspectrum.org/infoseekid.cgi?id=0027985



Auf der Chaos Construction wurde eine unbekannte Version von Robin Hood (1991 programmiert) gezeigt. Flimmernde Sprites und rollende Köpfe a la Barbarian überzeugen mich nicht so ganz. Die Programmierer waren unklug diese Version zu CCC einzureichen. Da muss noch viel verbessert werden. Wenigstens ist das Spiel nicht herunterladbar.



Galaxian III ist eines der Spiele von World XXI Soft Inc, die auf der Chaos Construction vorgestellt wurden (geschrieben bereits im Jahr 1992) und den dritten Platz belegte.
Leider ist der Download noch nicht verfügbar, nicht mal auf der Webseite von World XXI Soft Inc.



Imola G1, das dritte Spiel (Platz 4) schaut sehr interessant aus. Ein 3D Motorradrennen. Leider ist auch dieses Spiel noch nicht verfügbar gemacht worden.
Diese frühen Werke zeigen wie stark sich Coder verbessern können. „Carlos Michelis“ ist dann das Highlight.

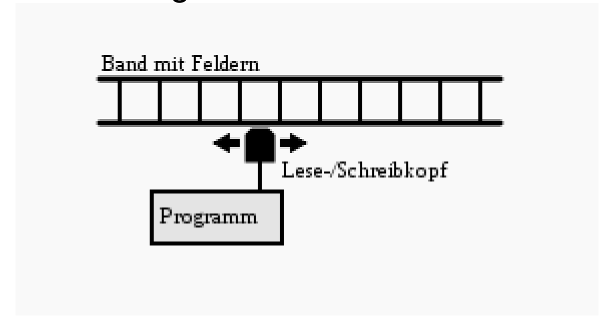
Die Geschichte der Computersimulationen auf dem ZX Spectrum

Die Reihe gibt Einblick in die Geschichte der Geschichte der Computersimulationen und lässt diese anhand von einfachen BASIC-Programmen auf dem ZX Spectrum wieder lebendig werden.

Folge 1: Turingmaschine und fleißige Biber

Am 12.11.1936 veröffentlichte Alan Turing einen Artikel zu einer Maschine, um die Berechenbarkeit von Funktionen zu zeigen. Sie sollte so einfach wie möglich konstruiert und dennoch in der Lage sein, alle bekannten Algorithmen abzuarbeiten. Der Originalartikel ist hier herunterladbar http://www.wolframscience.com/prizes/tm23/images/Turing.pdf und die Maschine wird nach ihrem Erfinder „Turingmaschine" genannt. Sie besteht im Original aus einem endlos langen Papierstreifen, auf dem sich Zeichen und Ziffern befinden. Die Turingmaschine beherrscht ausschließlich die Operationen „Lesen", „Schreiben" sowie den "Schreib-Lese-Kopf bewegen" und ist damit nach der Church-Turing-Hypothese in der Lage, alle Probleme zu lösen, die ein Computer oder ein Mensch lösen kann. Sämtliche mathematische Grundfunktionen wie Addition oder Subtraktion lassen sich mit Turingmaschine durchführen und darauf aufbauend dann auch alle komplexen Algorithmen der Computerprogramme.

Die Maschine besteht lediglich aus drei Teilen: dem oben erwähnten unendlichen Papierstreifen, einem Schreib-Lese-Kopf und einer Steuereinheit. Der Papierstreifen dient als serieller Speicher, in den der Schreib-Lesekopf einzelne Zeichen lesen und schreiben kann. Die Steuereinheit enthält das Programm zur Verarbeitung der Daten.



Aufbau der Turingmaschine

Die Turingmaschine liest die auf dem Streifen befindliche Eingabe mittels des Schreib-Lese-Kopfes und verändert sie entsprechend der Anweisungen im Programm. Anschließend bewegt sich der Schreib-Lese-Kopf ein Feld nach links oder rechts oder bleibt stehen. Welches Zeichen geschrieben und welche Bewegung ausgeführt wird, hängen sowohl vom Programm als auch vom Zustand ab, in dem sich die Turingmaschine befindet. Die Zustände werden durch Funktionen definiert, am Anfang befindet sich die Maschine im Startzustand und geht bei jedem Schritt in einen neuen Zustand über. Einzelne Zustände können mehrmals durchlaufen werden, außerdem kann man bestimmte Zustände als Endzustände definieren, bei deren Erreichen die Maschine stehen bleibt.

Die Turingmaschine auf dem ZX Spectrum

Das unten stehende BASIC-Programm konstruiert eine Turingmaschine auf dem ZX Spectrum.

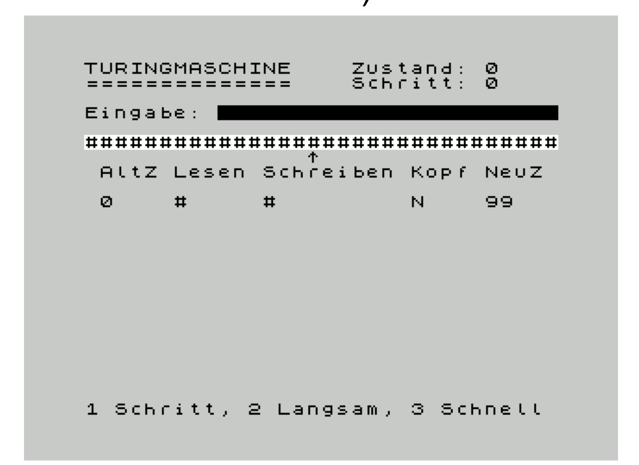
```
10 REM TURINGMASCHINE
15 REM Version 4
20 REM 2012 Wilko Schroeter
30 REM ********************
50 REM Aufbau der
Turingmaschine
60 REM Alter Zustand
70 REM Lesen
80 REM Schreiben
90 REM Kopf bewegen -1-
links,1-rechts,0-keine
Bewegung
100 REM neuer Zustand 99-
Halt
200 DIM p(5,100): REM
Programmspeicher
210 LET
d$="########################
########"
220 LET
f$="                       "
300 REM Init
310 LET az=0: REM Alter
Zustand
320 LET l=-1: REM Lesen
330 LET s=-1: REM Schreiben
340 LET k=1: REM
Kopfposition
350 LET nz=0: REM Neuer
Zustand
360 LET offset=15
370 LET schritt=0: REM
Schrittzaehler
1000 REM Grafische
Darstellung
1010 CLS
1020 PRINT AT
0,0;"TURINGMASCHINE"
1030 PRINT AT
1,0;"=============="
1042 PRINT AT 3,0;"Eingabe:
"; INVERSE 1;f$
1045 GO SUB 4900
1050 PRINT AT 7,1;"AltZ
Lesen Schreiben Kopf NeuZ"
2000 REM Programm lesen
2010 GO SUB 5000
2020 FOR i=1 TO z
2030 PRINT AT 8+i,1;p(1,i)
2040 IF p(2,i)>=0 THEN PRINT
AT 8+i,6;p(2,i)
2045 IF p(2,i)<0 THEN PRINT
AT 8+i,6;"#"
2050 IF p(3,i)>=0 THEN PRINT
AT 8+i,12;p(3,i)
2060 IF p(3,i)<0 THEN PRINT
AT 8+i,12;"#"
2070 IF p(4,i)=-1 THEN PRINT
AT 8+i,22;"L"
2080 IF p(4,i)=1 THEN PRINT
AT 8+i,22;"R"
2090 IF p(4,i)=0 THEN PRINT
AT 8+i,22;"N"
2100 PRINT AT 8+i,27;p(5,i)
2110 NEXT i
3000 REM Dateneingabe
3010 INPUT "Daten: ";i$
3015 PRINT AT 3,9; INVERSE
1;f$
3020 PRINT AT 3,9; INVERSE
1;i$
3024 IF i$<>"" THEN FOR i=1
TO LEN i$: LET d$(i+offset
TO i+offset)=i$(i TO i):
NEXT i
3030 GO SUB 4900
3050 REM Warteschleife
3060 PRINT #1;"1 Schritt, 2
Langsam, 3 Schnell"
3070 LET a$=INKEY$: IF NOT
(a$="1" OR a$="2" OR a$="3")
THEN GO TO 3070
3080 IF a$="1" THEN PAUSE 0
3090 IF a$="2" THEN PAUSE
```

```
100
3200 REM Hauptprogramm
3210 GO SUB 4500: REM
Programmverarbeitung
3220 GO SUB 4900: REM Band
zeichnen
3230 IF a$="1" THEN PAUSE 0
3240 IF a$="2" THEN PAUSE
100
3245 IF az=99 THEN STOP
3250 GO TO 3200
4500 REM
Programmverarbeitung
4505 LET c$=d$(k+offset TO
k+offset)
4506 IF c$="#" THEN LET c=-
1: GO TO 4510
4507 LET c=VAL c$
4510 LET f=0: REM
Flagvariable zum
Schleifenverlassen
4515 FOR i=1 TO z
4520 IF p(1,i)=az THEN IF
p(2,i)=c THEN IF f=0 THEN
FOR j=1 TO z: PRINT AT
8+j,0;" ": NEXT j: PRINT AT
8+i,0; INVERSE 1; FLASH
1;">": LET d$(k+offset TO
k+offset)=STR$ (p(3,i)): LET
k=k+p(4,i): LET az=p(5,i):
LET f=1: LET
schritt=schritt+1: IF
p(3,i)<0 THEN LET d$(k-
p(4,i)+offset TO k-p(4,i)
+offset)="#"
4550 NEXT i
4560 RETURN
4895 STOP
4900 REM Band zeichnen
4905 PRINT AT
6,0;"
          "
4910 PRINT AT 6,k+offset-
1;"^"
4930 PRINT AT 5,0; BRIGHT
1;d$
4940 PRINT AT 0,18;"Zustand:
";az
4945 PRINT AT 1,18;"Schritt:
";schritt
4950 RETURN
4999 STOP
5000 REM Programmcode
5005 REM 1 Zustand
5007 LET z=1: REM Anzahl der
Programmzeilen
5010 LET p(1,1)=0: LET
p(2,1)=-1: LET p(3,1)=-1:
LET p(4,1)=0: LET p(5,1)=99
9999 RETURN 9999 RETURN
```

Zuerst wird die Eingabe der Zeichen verlangt, die sich auf dem Papierstreifen befinden sollen. Anschließend lässt sich wählen, ob die Turingmaschine schnell, langsam oder im Einzelschrittmodus ausgeführt werden soll. Das eigentliche Programm für die Turingmaschine befindet sich ab der Programmzeile 5000 im BASIC-Listing. Jede Zeile enthält fünf Parameter:

1. Alter Zustand: in welchem Zustand muss sich die Turingmaschine befinden, damit die Zeile ausgeführt wird,
2. Lesen: welches Zeichen muss vom Papierband gelesen werden, damit der Rest der Zeile ausgeführt wird,
3. Schreiben: welches Zeichen soll der Schreib-Lese-Kopf auf das Band schreiben,
4. Kopf: wie soll sich der Schreib-Lese-Kopf anschließend bewegen (links, rechts, stehen bleiben),
5. neuer Zustand: in welchen Zustand soll die Turingmaschine anschließend springen (Zustand 99 bedeutet Halt und

damit den Endzustand).



Simulation einer Turingmaschine auf dem ZX Spectrum

Standardmäßig beginnt die Turingmaschine im Zustand 0 und liest das erste Zeichen auf dem Band. Das im obigen Listing enthaltene Programm für die Turingmaschine besteht aus nur einer Zeile. Diese besagt, dass, wenn

1. sich die Turingmaschine im Zustand 0 befindet und
2. ein leeres Zeichen (durch „#“ gekennzeichnet) vom Band gelesen wird,
   dann
3. wird ebenfalls ein leeres Zeichen auf das Band geschrieben,
4. der Schreib-Lese-Kopf nicht bewegt und
5. die Turingmaschine in den Zustand 99 (= Halt) versetzt.

Die Turingmaschine macht bei diesem Programm also nichts anderes als ein leeres Zeichen vom Band zu lesen und anschließend einfach anzuhalten.

Beispiel: Inkrement

Das nächste Programm ist schon etwas anspruchsvoller: es erhöht die auf dem Band befindliche Binärzahl um 1. Das dazugehörige Programm besteht aus 9 Zeilen und benötigt 2 Zustände:

```
5000 REM Beispielprogramm
Inkrement
5005 REM 2 Zustaende
5007 LET z=9: REM Anzahl der
Programmzeilen
5010 LET p(1,1)=0: LET
p(2,1)=0: LET p(3,1)=0: LET
p(4,1)=1: LET p(5,1)=0
5020 LET p(1,2)=0: LET
p(2,2)=1: LET p(3,2)=1: LET
p(4,2)=1: LET p(5,2)=0
5030 LET p(1,3)=0: LET
p(2,3)=-1: LET p(3,3)=-1:
LET p(4,3)=-1: LET p(5,3)=1
5040 LET p(1,4)=1: LET
p(2,4)=0: LET p(3,4)=1: LET
p(4,4)=-1: LET p(5,4)=2
5050 LET p(1,5)=1: LET
p(2,5)=1: LET p(3,5)=0: LET
p(4,5)=-1: LET p(5,5)=1
5060 LET p(1,6)=1: LET
p(2,6)=-1: LET p(3,6)=1: LET
p(4,6)=0: LET p(5,6)=99
5070 LET p(1,7)=2: LET
p(2,7)=0: LET p(3,7)=0: LET
p(4,7)=-1: LET p(5,7)=2
5080 LET p(1,8)=2: LET
p(2,8)=1: LET p(3,8)=1: LET
p(4,8)=-1: LET p(5,8)=2
5090 LET p(1,9)=2: LET
p(2,9)=-1: LET p(3,9)=-1:
LET p(4,9)=0: LET p(5,9)=99
9999 RETURN
```

Ausgehend von solchen einfachen mathematischen Operationen wie Addition, Subtraktion oder Multiplikation mit den benötigten 2 Zuständen lassen sich auch alle (von Menschen erdachten) komplexen mathematischen Algorithmen mittels der Turingmaschine ausführen. 2007 bewies ein Alex Smith aus Birmingham, dass eine Turingmaschine mit 2 Zuständen

tatsächlich dermaßen universell einsetzbar ist und gewann damit ein Preisgeld von 25.000 $ (http://www.heise.de/newsticker/meldung/97991).

Beispiel: Fleißige Biber

```
TURINGMASCHINE        Zustand: 99
==============        Schritt: 14

Eingabe: 101111

#################110000############
                 ↑
 AltZ  Lesen  Schreiben  Kopf  NeuZ

  0      0       0        R     0
  0      1       1        R     0
  0      #       #        L     1
  1      0       1        L     2
  1      1       0        L     1
  1      #       1        N     99
  2      0       0        L     2
  2      1       1        L     2
 >2      #       #        N     99


9 STOP statement, 3245:2
```

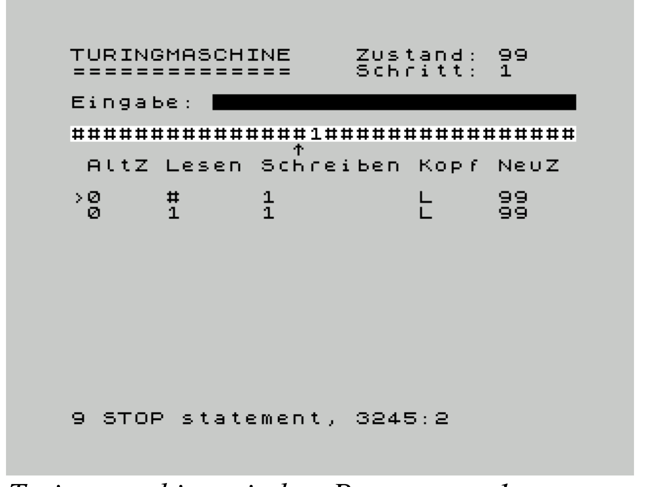
Turingmaschine mit dem Programm „Inkrement“

Ein Problem, das bereits viele Mathematiker zu Überlegungen angeregt hat, ist die Frage, wie viele Einsen eine Turingmaschine auf das Band schreiben und anschließend anhalten kann. Diese Art von Programmen nennen sich „fleißige Biber“ („Busy Beaver“). 1962 entwickelte Tibor Radó aus Ungarn die sogenannte Radó-Funktion, die angibt, wie viele Zeichen eine Turingmaschine maximal bei einer gegebenen Anzahl von Zuständen schreiben kann. Für den Einserbiber mit nur einem Zustand sieht das Programm noch sehr übersichtlich aus:



```
5000 REM Fleissige Biber
5005 REM 1 Zustand
5007 LET z=2: REM Anzahl der Programmzeilen
5010 LET p(1,1)=0: LET p(2,1)=-1: LET p(3,1)=1: LET p(4,1)=-1: LET p(5,1)=99
5020 LET p(1,2)=0: LET p(2,2)=1: LET p(3,2)=1: LET p(4,2)=-1: LET p(5,2)=99
9999 RETURN
```

Es besteht aus lediglich zwei Zeilen und ist in der Lage, gerade einmal eine „1“ auf das Band zu schreiben und dann anzuhalten:



Turingmaschine mit dem Programm „1er Biber“

Mit dem 2er Biber wird es bereits möglich, mit 6 Schritten 4 Einsen auf das Band zu schreiben und zu stoppen. Für 3 Zustände lassen sich mit 11 Schritten maximal 6 Einsen schreiben, bei 4 Zuständen beträgt die Maximalzahl der Einsen 13 bei 107 Schritten.

Interessant werden eigentlich erst die 5er Biber, weil bis heute noch nicht der Nachweis gelungen ist, wie viele Einsen eine Turingmaschine mit 5 Zuständen genau schreiben kann, da dazu alle

möglichen Turingmaschinen mit 5 Zuständen betrachtet werden müssten. Die Gesamtzahl beträgt übrigens 22^{10} = 26.559.922.791.424, d. h., wenn es gelingen würde, jede Sekunde eine Turingmaschine mit 5 Zuständen zu generieren und auf die Anzahl der geschriebenen Einsen abzutesten, würde dies ca. 840.000 Jahre in Anspruch nehmen! Die heutigen Rekordhalter Heiner Marxen und Jürgen Buntrock entwickelten 1989 eine Turingmaschine mit 5 Zuständen, die 4098 Einsen auf das Band schreibt und dafür 47.176.870 Schritte braucht.

```
TURINGMASCHINE        Zustand:  99
==============        Schritt:  107

Eingabe:

#####1#111111111111#############
      ↑
 AltZ Lesen Schreiben Kopf NeuZ

 0    #     1         R    1
 0    1     1         L    1
 1    #     1         L    0
 1    1     #         L    2
>2    #     1         N    99
 2    1     1         L    3
 3    #     1         R    3
 3    1     #         R    0


9 STOP statement, 3245:2
```

Turingmaschine mit dem Programm „4er Biber“

Beim 6er Biber steht der Rekord seit 2010 bei $3{,}18 \times 10^{10566}$ Einsen, erzielt von Pavel Kropitz. Vielleicht gelingt es jemandem mittels ZX Spectrum und obiger Simulation ihn zu übertrumpfen?

Wilko Schröter

Adventurelösung: Retarded Creatures and Caverns

Hallo liebe Adventurefreunde!

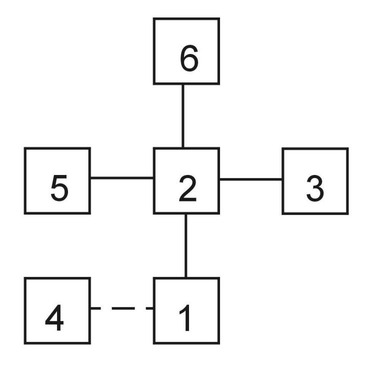
Wie ihr es von uns schon seit langer Zeit gewohnt seid, heute wieder etwas für die Adventurefreunde unter den Usern. Zur Lösung haben wir uns heute "Retarded Creatures and Caverns" ausgesucht. Das Programm selbst ist realtiv kompakt was den Lösungsweg betrifft, kommt man doch mit 6 (sechs!!!:) Locations durch das ganze Adventure. Trotzdem sind einige recht vertrackte Rätsel zu lösen und die Lösung liegt nicht immer sogleich auf der Hand. Kombinieren ist also nicht verboten und manchmal unbedingt nötig, will man erfolgreich bestehen. Im Programm trifft man natürlich wieder auf den in nahezu jedem Adventure vorkommenden Drachen und auch sonst liegt der eine oder andere Gegenstand herum, der nützlich sein könnte. Doch kommen wir ohne lange Vorrede jetzt gleich einmal zu den Locations unseres beiliegend abgedruckten Planes:

01) Standing before the great oak door of Castle Toidi / stone, tatty map, small pouch, sheet of instructions, gold coin, torches, large brass key
02) In a very large ante-chamber / large Dragon, metal handle
03) Wandering in a downward sloping tunnel -> middle of a circular room / rubbish, wand - sword, burger
04) Standing in the middle of a small pouch
05) Crawling in a tiny tunnel -> standing in the middle of a small atic / large golden eagle, box
06) In a tiny and very dark alcove / tiny gold coin

Das waren also schon die Locations inklusive Gegenstände, die wir auf unserem Lösungsweg betreten werden. Sicherlich gibt es noch eine ganze Menge mehr, aber die brauchen wir

nicht. Nun also der Lösungsweg, heute einmal wegen der Kürze der Lösung ein wenig ausführlicher kommentiert:

Retarded Creatures and Caverns



(c) 2012 by Harald R. Lack, Möslstraße 15 a, 83024 Rosenheim
und Hubert Kracher, Schulweg 6, 83064 Raubling

Examine door (die Beschreibung des Schlosses, vor dem wir stehen, erscheint), examine stonework (wir entdecken einen losen Stein), get stone, examine lintel (wir sehen eine Flechte), rub lichen (eine Inschrift wird sichtbar - mitunter muß man diesen Schritt mehrfach wiederholen bis es klappt), read inscription (wir erfahren, wie man das Schloßtor öffnet - die fehlenden Worte sind Tail & Ass), read name (wir erfahren den Namen des Steinmetzes), scrape boots (das sollten wir nicht vergessen bevor wir weitermachen), examine scraper (bei der Untersuchung stellen wir fest, daß er die Form eines Affen hat - wenn wir den Schwanz anheben, wird sich das Tor öffnen, aber wir brauchen noch etwas um den Schwanz zu arretieren), wedge tail (mit dem Stein), go north (wir werden von einem Tentakel aus dem Moor ergriffen und eine Stimme fragt uns, wer hier passieren will), Bulbo (das war der richtige Name - wir kommen frei und erhalten noch eine Goldmünze, die uns in die Hand gedrückt wird), N (wir gelangen in eine Ante-chamber - fast jedes Adventure hat so eine Location), examine chamber (wir entdecken einen Griff oder eine Kurbel an der nördlichen Wand), examine handle (wir finden heraus, wie sie funktioniert), examine dragon (er sitzt auf einem Haufen Gold und verdeckt zum Teil einen Tunnel nach Westen), worn (wir wollen mal nachsehen, was wir so anhaben - darunter ist eine Weste aus Fäden), unravel vest (wenn wir die Weste entwirren erhalten wir einen Faden den man um etwas binden könnte), tie string (jetzt kommt die Frage, worum wir den Faden binden wollen), to handle, pull string (im Osten erscheint eine dunkle Öffnung die wir wohl nur mit einer Lichtquelle erkunden können), S (wieder raus aus dem Schloss und die Schlossmauern erkunden), examine wall (hier gibt es doch tatsächlich zwei der in früheren Tagen so beliebten Fackeln und es ist gut dass es zwei sind. Warum? Ihr werdet es sogleich merken), get torch, N, E (in der Dunkelheit wird uns unsere Fackel gestohlen also gehen

wir raus und holen die zweite), W, S, get torch (während dieser Zeit wird uns dann gleich auch noch von einem Kobold unser Gold gestohlen. Aber das soll uns jetzt nicht weiter kümmern. Wie ihr sehen werdet bekommen wir es später wieder zurück. Ein Kobold wird uns auch noch fragen, ob wir das Adventure als Warrior oder Magician spielen wollen. Wie bei vielen Dingen im Leben ist es völlig egal wie wir antworten), N, E (zurück ins Schloss - in der dunklen Öffnung finden wir einen Haufen Unrat), examine rubbish, search rubbish (bis wir ein Schwert ODER einen Zauberstab finden - dann noch weitersuchen, bis wir auch noch einen Hamburger finden - anscheinend war auch schon in damaliger Zeit Fastfood ein Thema), get sword oder wand, get hamburger, W (zurück in die ante-chamber und den Drachen gefüttert), feed dragon (mit dem Hamburger worauf er in Schlaf fällt), S (wieder einmal außerhalb des Schlosses - wir legen jetzt alles ab, was wir bei uns haben), climb into pouch, feal east (wir entdecken eine kleine Kiste), get chest, climb out of pouch, get stone (den unter dem Schwanz des Affen), smash chest (mit Hilfe des Steines der uns nach getaner Arbeit aus der Hand gleitet und ins Moor rollt. Unter den Trümmern der Kiste ist auch ein Schlüssel), get key (durch das Wegnehmen des Steines schließt sich die Eingangstüre des Schlosses. Da wir wieder hinein müssen ist es sinnvoll ein wenig zu warten), wait (ein kleiner Kobold wird irgendwann erscheinen und uns die Frage aller Fragen stellen: Wie ist der Name des Steinmetzes?), Nardo, und er lässt uns wieder ins Schloss zurück), N, W (in dem Tunnel, in den wir einen Adler vorfinden, der durch einen Ball und eine (die Kette ist mit einem Vorhängeschloß gesichert), unlock padlock (der Adler zeigt uns seine Dankbarkeit indem er uns eine Schachtel gibt bevor er davonfliegt), examine box (wir sehen einen Knopf und eine ausziehbare Antenne), extend aerial, E (zurück in der ante-chamber - der Drache ist jetzt aufgewacht), press button (ein hilfreicher Geselle erscheint), press button (Bulbo erscheint - vorausgesetzt wir haben seine Stiefel nicht ruiniert, wird er den Drachen für uns beseitigen und wir kommen wieder an das Gold), get gold, pull handle (damit öffnet sich die Tür in der Nordwand), N (in einem Alkoven, wo unser Gold sich befindet), get gold, S (wir sind im Hof - die Eingangstüre des Schlosses ist mal wieder zu), kick door (und schon ist sie auf), S (wieder werden wir von einem Tentakel ergriffen, aber das ist keine Problem, da wir das Gold und das Goldstück bei uns haben - Retarded Creature and Caverns ist gelöst.

Das war es auch schon in diesem zugegeben etwas verwirrenden Adventure. Manchmal liegt die Würze eben in der Kürze. Aber wollen wir an dieser Stelle nicht groß philosophieren. Stürzen wir uns auf das nächste Adventure. Soviel von uns und bis demnächst an dieser Stelle ......

English summary

Dear adventure friends,

some of you may remember the game “Retarded Creatures and Caverns”. This

is a very curious adventure as we think for our solution takes place in only six locations. Nevertheless it contains some strong to solve riddles and the player has to combine a lot, which makes the game also interesting. During our journey through the program, we will see, that some heavy thinking is necessary to make a successful progress. But lets make it happen together if you like following our map and solution printed above.

New „40 best procedures“

(C) Kolotov Sergei g.Shadrinsk,
SerzhSoft, July, 1997.

In 1992, on pages ZX-REVIEW published an abridged translation of the book Dzh.Hardmana and E. Hyuzon "top 40 procedures." The publication has caused the most enthusiastic responses of readers because many budding spectrum users just use it to finally able to overcome the difficulties "of the barrier machine code." But here is exactly five years, and old-timers were surprised ZX REVIEW notice in the first issue for 1997year before the pain familiar name.
All the numerous arguments so convincingly set forth INFORKOMOM, of course, largely correct. But in my (and not only my) opinion, they have not entitle the national magazine publishers to spend precious Page ZX-REVIEW print already published at the time of the material.
Instead of heading a "retro" is much more useful and relevant it would be enter the section "remake." And it print is not just the old materials, and consider them more deeply, to give examples more effective implementation programs and other procedures.
Get at least the same "40 procedures." Yes, this work is very useful for beginners. But slightly more experienced experts assembler will notice that the more on obemu reducible authors procedure to consistently it has more disadvantages more effectively implemented, has more "extra" commands ... Yes, some "Better procedures" are reduced more than twice! And: Comments are for by the listings, which is very difficult to understand "what it occurs when the procedure is executed at a certain stage. "Of course, a description Type "C register is copied into the register B", but here is the semantic burden, which stands behind it, understood is not always ...
It will be presented libraries, which integrates graphic procedures, recopied on a "forty-best." The memory footprint was reduced from 1444 bytes to 861 bytes! Each procedure commented in detail in the Listing her, so to speak - "is not on the spot." Listing and hex dump.

Many of the procedures in their work requires some precertain values - constants. Under these values are allocated a special area of memory addressable Tagged CONSTS.
In this case CONSTS indicates address 23296, but, of course, this address can be changed to any other. The length of the constants of 8 bytes. If any of the procedures, none of the constants does not change. Otherwise, would have to call them variables ...

The procedures that manipulate coordinates of points on the screen,

ostchet is in contrast to BASIC is not a bottom-up, and vice versa - top-down. Such a reading of coordinates is much more convenient and is used in many other computers. Now you can specify Y coordinate from 0 to 191 (instead of 175), ie it is possible to specify the coordinates of the dots on the screen that are in the bottom two rows to be allocated under the error message. When counting the same bottom-up maximum Y-coordinate is Y = 175, and to the lower of the two lines do not reach.

BRIEF DESCRIPTION OF PROCEDURES

1. ASRL_LF

- a procedure shifts the entire screen attributes (color) to the left. The right column is filled with an attribute of the cell to at CONSTS (23,296). Length procedure, 22 bytes (was – 23 bytes). -Translated by the address 62000 (HEX: # F230).

2. ASRL_RG

- translation of the whole screen attributes to the right. Left column filled with an attribute of CONSTS. The length of the procedure, 21 bytes (19 bytes). Located at 62022 (# F246).

3. ASRL_UP

- translation of the whole Screen attributes up. Bottom string attribute is populated from CONSTS. The length of the procedure, 19 bytes (21 bytes). Address: 62043 (# F25B).

4. ASRL_DN

- translation of the whole Screen attributes down. Upper attribute is to be filled out CONSTS. Length: 20 bytes (21 bytes). Address: 62062 (# F26E).

5. SSRL_LF

- translation of the whole display left one character (Graphics). The right column familiarity cleared. Length of 20 bytes (21 bytes). Address: 62082 (# F282).

6. SSRL_RG

- translation of the whole screen to the right by one character. The left column familiarity cleared. Length of 19 bytes (22 bytes). Address: 62102 (# F296).

7. SSRL_UP

- translation of the whole the screen up one character. The bottom line of familiarity is cleared. Length 55 bytes (68 bytes). Address: 62121 (# F2A9).

8. SSRL_DN

- translation of the whole screen down by one character. The top line of familiarity is cleared. Length 55 bytes (73 bytes). Address: 62176 (# F2E0).

9. PSRL_LF

- translation of the whole to the left by one pixel (Graphics). The right column of pixels is cleared. Length of 16 bytes (17 bytes). Address: 62231 (# F317).

10. PSRL_RG

- translation of the whole screen to the right by one pixel. The left column of pixels is cleared. Length 17 bytes (17 bytes). Address: 62247 (# F327).

11. PSRL_UP

- translation of the whole the screen up one line of pixels. The bottom line of pixels cleared. Length of 38 bytes (91

bytes). Address: 62264 (# F338).

12. PSRL_DN
 - translation of the whole screen down one line of pixels. The top row of pixels cleared. Length of 38 bytes (90 bytes). Address: 62302 (# F35E).

13. SCR_MRG
 - the merging of two images (graphics, on the principle of OR). Double-byte constant for CONSTS address must contain location, where the second picture in memory (overlay). The result is put on the screen. The length of the procedure, 17 bytes (before - 21 bytes). Address of accommodation: 62,340 (# F384).

14. SCR_INV
 - invert screen (graphics, on the principle of NOT). All the pixels change their meaning the opposite. the length of the procedure, 12 bytes (was - 18 bytes). Address: 62357 (# F395).

15. SINV_UD
 - inverting the symbol vertically. Arrow pointing upwards, is directed downward, and vice versa. In the double-byte variable at CONSTS must contain the address of a variable character. The length of the procedure, 20 bytes (so was). Address allocation: 62,369 (# F3A1).

16. SINV_LR
 - inverting the symbol horizontally. Arrow pointing left, is directed to the right, and vice versa. In the double-byte variable at CONSTS must contain the address of a variable character. The length of the procedure, 17 bytes (was - 19 bytes). Address of accommodation: 62,389 (# F3B5).

17. SROTATE
 - rotate the character in a clockwise direction at 90 degrees. In the double-byte variable at CONSTS must contain the address of a variable character. Length of 26 bytes (previously - 42 bytes). Address: 62406 (# F3C6).

18. ACHANGE
 - change the attributes of all symbols screen. Bit operation. In cell at CONSTS should contain the mask bits: Bits that are set in a mask remain in the attributes of the same, and Bits that are masked are zero - would have zero value and the attributes (operation AND (not "Y "!)). The cell at CONSTS +1 bytes must be included bits of which will be introduced in all the attributes of the screen, ie, if this byte is a bit turned on, it will be installed in all attributes (OR operation). Length procedure, 16 bytes (21 bytes). Address: 62432 (# F3E0).

19. AREPLC
 - search the attributes of a specific value and replacing each found a new attribute value. In cell at CONSTS should contain a byte value to be replaced (that look). In cell at CONSTS +1 must be the value of the substitute bytes (the change). The length of the procedure, 18 bytes (22 bytes). Address: 62448 (# F3F0).

20. PAINT
 - fill a specific area of the screen, bounded by a line of pixels (fill). The starting point is given by putting her on the X at CONSTS, and the coordinates of Y - at CONSTS +1. If the Y coordinate greater than 191 or the point at these

coordinates is already installed, the program urgently interrupted. This procedure does not moveable due to procedure calls POINT. When shading is actively uses the stack - it remembers the coordinates of the fill lines. When painted over a large region of complex shape, it is necessary and more space in RAM - between the end BASIC-program and the address set by the operator CLEAR (The contents of the system variable RAMTOP). If memory space is not enough, it may crash. The procedure takes 88 bytes, and together with the procedure POINT – 123 bytes, which is more than twice less than the length of procedure 1992 (263 bytes!) Address PAINT: 62,466 (# F402).

21. POINT

- address calculation point in the screen given the coordinates and status this point (ON / OFF). Attention! This procedure can only be used from native code (Start of BASIC nothing will). Before the call to set in the register E coordinate X (0 .. 255), and in the case D - coordinate Y (0 .. 191) tested the point. The output of the procedure set in the register pair HL byte address on the screen, which is the point, and in case C - the point in this mask byte (one bit set in unit). Depending on whether including the points or not, a flag is set zero: Z - a point not included, NZ - point is included. If the point is established (visible) then register A (accumulator) is identical in meaning with the register C, and if the point is not set, then A is reset. Register B is always at the exit from the procedure is zero. The length of the procedure, 35 bytes (the original took would be about 70 bytes). Address allocation: 62554 (# F45A).

22. PFIGURE

- construction of any pre-defined shapes (Pattern) on the screen. Coordinates of the initial (starting) points are given the same procedure PAINT. Template is given in a string variable BASIC A $ (can change to any other, slightly adjusting assembly listing or dump). A string of characters has the following format (a little differs from the original):

"5" - to reduce the X-coordinate
"6" - to increase the Y-coordinate (The count goes from top to bottom)
"7" - to reduce the Y-coordinate
"8" - to increase the X-coordinate
"0" - put the point.

Any other characters are ignored. If the string variable does not exist or does not contain any information, then the program stops its work. Control the output of the initial Y-coordinat not, because part of the figure can still be seen. Therefore, checking the output for off-screen to put in myself cycle of forming a pattern. Ability to 'wrap-round' is preserved, ie still, when the output X-coordinates for the left side of the screen, the pattern appears to the right, and vice versa.
The procedure is not moveable. Length PFIGURE: 98 bytes, and is used together with routine POINT - 133 bytes, which is still much smaller than the original (196 bytes). Address: 62589 (# F47D).
If the "open" procedure call POINT, you PFIGURE can be moved and will occupy about 125 bytes!

23. PSCALER

- copy of the screen to another area of the same screen with a possible increase up to X and / or Y. Constants are used:

Address Name Comment
CONSTS X1_OLD one of the two initial X-coordinate
CONSTS+1 Y1_OLD one of the two initial Y-coordinate
CONSTS+2 X2_OLD one of the two initial X-coordinate
CONSTS+3 Y2_OLD one of the two initial Y-coordinate
CONSTS+4 XSCALE magnitude increase in X
CONSTS+5 YSCALE magnitude of increase in Y
CONSTS+6 X_NEW coordinates of the upper-left corner of the screen,
CONSTS +7 Y_NEW which is made up.

Coordinates of the initial rectangle for copy set in the constants X1_OLD, Y1_OLD, X2_OLD, Y2_OLD, moreover, may arranged in any order. The procedure itself will determine the smallest and largest coordinates.
Emergency exit procedures occurs in the following cases:

1. XSCALE = 0 - zoom scale on X is zero
2. YSCALE = 0 - zoom scale the Y equals zero
3. Y_NEW> 191 - the new coordinate Y beyond the screen
4. Y1_OLD> 191 - the old coordinates that Y1 beyond eq wound
5. Y2_OLD> 191 - the old coordinates Y2 that goes beyond the equivalence wound.

Like the original, the program does not control that would check the possibility of placing new picture on the screen. If this does not work, you may crash.
When performing the procedure at first throws a stack bit image of the copied rectangle screen and then draws it to the new location, increasing if necessary. Therefore, if on the stack is not enough space, then as in the procedure PAINT, can occur hangs, reset, Error ...
If you want to copy a piece of the screen had the same size as a given, then necessary to establish the scale of 1:1 - add a constant XSCALE and YSCALE on yedinichku. When you double the amount there should be dvoechki, and so on ...
The procedure is not moveable from the use of sub POINT. PSCALER occupies 174 bytes, and with POINT – 209 bytes. In any case, it is much smaller than the original – 335 B! Address allocation: 62,687 (# F4DF).

So, here you are familiar with and new implementation of the "better procedures". But I advise not to have illusions - some of the procedures it should be possible to reduce more ... True, this will inevitably have to sacrifice something: the speed performance, transferability and, finally, just the time spent. I hope that presented in this paper the program will be useful, in extreme cases - may be, they just touch up your thoughts ...
Beginners can try to compare the procedure in 1992 with new study principles for creating more effective programs, tools for the integration of many different procedures into one big library ... Experienced programmers are the same, may get a lot of fun, laughing mischievously over this work. But it is also Plus: amuse people – very the right thing! So I wish you all readers to enjoy

working with LOVED SPECCY!

40 New Best Routines (graphic);
(c) SerzhSoft, Shadrinsk, may-june, 1997; Old length: 1444 bytes new length: 861 bytes;

```
;-------------------------------------
ORG 62000; address assembly
;-------------------------------------
CONSTS EQU 23296; buffer address
constant (8 bytes)
; Shift attributes to the left (22 <=
23)

;ASRL_LF

LD DE, # 5800; DE = address of first
byte of the attribute LP_ASLF
LD H, D; DE copied in HL
LD L, E; and HL increased by one:
INC HL; HL = address of the second byte
of attributes
LD BC, # 001F; <line length attribute>
- 1
LDIR; shift of the attributes of the
left
LD A, (CONSTS); fill color after the
shift

LD (DE), A; set a new attribute
INC DE; the transition to the next line
below
LD A, D; if the attributes have already
run out,
CP # 5B; and we came upon a printer
buffer,
JR C, LP_ASLF; then STOP, otherwise
continue to shift
RET; exit procedures

; Shift attributes to the right (21 <=
23)
;ASRL_RG

LD DE, # 5AFF; address of the last byte
of attributes LP_ASRG
LD H, D; DE copied in HL -
LD L, E; the last byte of the line
attributes
DEC HL; the penultimate line of bytes
of attributes
LD BC, # 001F; <line length attribute>
- 1
LDDR; shift of the attributes of the
right
LD A, (CONSTS); fill color after the
shift
LD (DE), A; set a new attribute
DEC DE; the transition to the next line
from the top
BIT 3, D; if we are still in the
attributes,
JR NZ, LP_ASRG; then repeat the cycle
for the lyrics line
RET; exit procedures
;-------------------------------------
; Shift attributes up (19 <= 21)
;-------------------------------------
; ASRL_UP

LD HL, # 5820, and the address of
second-line attributes
LD DE, # 5800; address of the first
line of the
attributes
LD BC, # 02E0 3; move: 23 lines of 32
bytes
LDIR 3, shifting the bottom line up 23
LD A, (CONSTS); color to fill the
bottom line
LP_ASUP LD (DE), A; set a new attribute
INC E; if you have filled the whole
last line
JR NZ, LP_ASUP; (E = 0), interrupts the
cycle
RET; exit procedures
;-------------------------------------
; Shift attributes down (20 <= 21)

; ASRL_DN

LD HL, # 5ADF; address of the end of
the second line below
LD DE, # 5AFF; address of the end of
the bottom line
LD BC, # 02E0 3; move: 23 lines of 32
bytes
LDDR 3, shifting the line down the top
23
LD A, (CONSTS); color to fill the top
line
LP_ASDN LD (DE), A; set a new attribute
DEC E; if you come down to the very
first byte
JR NZ, LP_ASDN; field attributes (E =
0) then STOP
LD (DE), A; and set the byte
RET; exit procedures
;-------------------------------------
; ; Shift left one character (20 <= 21)
;-------------------------------------
; SSRL_LF

LD DE, # 4000; top of graphics
```

```
LP_SSLF LD H, D; address of the first
LD L, E; byte line
INC HL; address of the second byte of
the line
LD BC, # 001F; how many bytes to shift
LDIR; line shift to the left by 1 byte
XOR A; resets the battery and have
brought
LD (DE), A; the last (right) bytes line
INC DE; the transition to the next line
(below)
LD A, D; if the attributes
CP # 58; "not yet seen"
JR C, LP_SSLF; then repeat the cycle
for the lyrics.
line
RET; exit procedures
;--------------------------------------
; Right shift by one character (19 <=
22)
;--------------------------------------
; SSRL_RG
LD DE, # 57FF; the last byte in the
field of graphics
LP_SSRG LD H, D; address of the last
byte
LD L, E; current line
DEC HL; address penultimate bytes
LD BC, # 001F 3; shift: 31 bytes
LDDR; shift of the line graphs to the
right
XOR A; clear the battery and then

LD (DE), A; first (left) bytes of the
current line
DEC DE; the transition to the next line
above
BIT 6, D; if we did not "come across"
on the ROM
JR NZ, LP_SSRG; then continue to twist
the loop
RET; exit procedures
;--------------------------------------
; Shift up one character (55 <= 68)
;--------------------------------------
; SSRL_UP
LD DE, # 4000; top display area LP_SSU1
PUSH DE; stores the address line on the
stack
LD BC, # 0020 3 line - 32 bytes
LD A, E; The register address is DE
ADD A, C; top line. The Register
LD L, A; HL to get the address
LD A, D; line lying below with step 8.
JR NC, GO_SSUP; For this sensitive E
pribav
ADD A, # 3 August, trolled 32 and fills
in L. If ProGO_SSUP LD H, A; emanated
overflow, then H = D +8
LDIR 3; transfer line (32 bytes)
POP DE; restore the address of the
start line
LD A, H; check: Is not it time we
closed
CP # 58 3; glyatsya? (Transferred all
23 series)
JR NC, LP_SSU2; if so, the transition
to clean
INC D
;--------------------------------------
LD A, D; DOWN_DE
AND # 07; standard sequence
JR NZ, LP_SSU1; teams to go on line
LD A, E; down in the display area
ADD A, # 20, (to register DE)
LD E, A;
JR C, LP_SSU1; input: DE - address line
LD A, D; output: DE - address line
below
SUB # 08; battery is used
LD D, A;
JR LP_SSU1
;-------------------------------------
LP_SSU2 XOR A; cleaning battery
LP_SSU3 LD (DE), A; and with his help -
INC E; cleaning of one line image
JR NZ, LP_SSU3 3 total: 32 bytes
LD E, # E0; move to the next
INC D; (lower) line image
BIT 3, D; filled the entire last
series?
JR Z, LP_SSU2; if not, continue to fill
RET; exit procedures
;--------------------------------------
; Shift down by one character (55 <=
73)
;--------------------------------------
; SSRL_DN
LD DE, # 57FF; address of the last byte
of graphics
LP_SSD1 PUSH DE; store the address of
the end of the line
LD BC, # 0020, the length of one line
image
LD A, E; in register HL
SUB C; we address
LD L, A; end of the line
LD A, D; overlying
JR NC, GO_SSDN; initial steps
SUB # 08; of 8 pixels (lines): GO_SSDN
LD H, A; HL = from copy; DE = where
LDDR; transfer line graphs
POP DE; restore address the end of the
line
BIT 6, H; if we are no longer in the
screen,
JR Z, LP_SSD2; then go to clean up
LD A, D
```

```
;---------------------------------
DEC D; UP_DE
AND # 07; standard sequence
JR NZ, LP_SSD1; teams to go on line
LD A, E; up in the display area
SUB # 20; (to register DE)
LD E, A;
JR C, LP_SSD1; input: DE - address line
LD A, D; output: DE - address line
above
ADD A, # 08; battery is used
LD D, A;
JR
LP_SSD1 ;-----------------------------
---;
LP_SSD2: XOR A; cleaning battery
LP_SSD3: LD (DE), A; cleaning of one
DEC E; image line:
JR NZ, LP_SSD3 3 (31 bytes)
LD (DE), A; clear the very first byte
of the line
LD E, # 1F; move to the next (upper)
DEC D; through a number of the eight
lines
BIT 6, D; we have not got the ROM?
JR NZ, LP_SSD2; if not, then clear the
next
RET; exit procedures
;---------------------------------------
; Left shift by one pixel (16 <= 17)
;---------------------------------------
; PSRL_LF
LD HL, # 57FF; address of the last byte
of graphics
LP_PSL1: OR A; reset the carry flag CF
LD B, # 20 March, in the same line - 32
bytes
LP_PSL2 RL (HL); CF <- [Sliding bytes]
<-CF (left)
DEC HL; go to the previous byte line
DJNZ LP_PSL2; shift cycle for one line
BIT 6, H; we are still in the screen?
JR NZ, LP_PSL1; if so, shift the trail.
line
RET; exit procedures
;---------------------------------------
; Right shift by one pixel (17)
;---------------------------------------
; PSRL_RG
LD HL, # 4000; address the first byte
of graphics
LD C, # C0 3; shift - 192 lines
LP_PSR1: OR A; CF = 0 for an empty
column on the left
LD B, # 20, the number of bytes in one
line
LP_PSR2 RR (HL); shift one byte to the
right
INC HL; next byte of image line
DJNZ LP_PSR2 3, we shift the whole line
- 32 bytes
DEC C; decrease the counter lines
JR NZ, LP_PSR1; if you have moved all
the lines, then STOP
RET; exit procedures
;---------------------------------------
; Upward shift by one pixel (38 <= 91)
;---------------------------------------
; PSRL_UP
LD DE, # 4000; address of the beginning
of graphics (verh. line) LP_PSU1 LD H,
D; copied the address of the beginning
LD L, E; line graphs in HL
LD BC, # 0020, the size of a single
line
INC H
;---------------------------------
LD A, H; DOWN_HL
AND # 07; standard sequence
JR NZ, GO_PSUP; teams to go on line
LD A, L; down in the display area
ADD A, C; (for the register HL)
LD L, A; (here ADD A, C instead of ADD
A, # 08)
JR C, GO_PSUP; Input: HL - address line
LD A, H; Output: HL - address line
below
SUB # 08; battery is used
LD H, A
;---------------------------------
;GO_PSUP
PUSH HL; store address the bottom line
LDIR; transfer images from the bottom-
up 140.
POP DE; DE - address the bottom line
LD A, H; we are still in the field of
graphics
CP # 58; or have stumbled upon the
attributes?
JR C, LP_PSU1; if still graphics, then
repeat
XOR A; zero out the battery and its
LP_PSU2
LD (DE), A; help clear the most
INC E; bottom line of the image
JR NZ, LP_PSU2; after the shift the
screen up
RET; exit procedures
;---------------------------------------
; Shift down by one pixel (38 <= 90)
;---------------------------------------
; PSRL_DN
LD DE, # 57FF; address of the last byte
of graphics
LP_PSD1 LD H, D; copied the address of
the last
LD L, E; byte line HL
LD BC, # 0020; width of one image line
```

```
LD A, H
;---------------------------------
DEC H; UP_HL
AND # 07; standard sequence
JR NZ, GO_PSDN; teams to go on line
LD A, L; up in the display area
SUB C; (for the register HL)
LD L, A; (SUB C here instead SUB # 08)
JR C, GO_PSDN; Input: HL - address line
LD A, H; Output: HL - address line above
ADD A, # 08; battery is used
LD H, A
;--------------------------------;
;GO_PSDN
PUSH HL; store address the top line
LDDR; tolerated a line from the top - down
POP DE; address the top line has become the current
BIT 6, H; not yet got into the ROM -
JR NZ, LP_PSD1; continue the cycle of transmission lines
XOR A; clear the battery and its
LP_PSD2: LD (DE), A; help - the top-line
DEC E; image after the shift
JR NZ, LP_PSD2; the entire screen down
LD (DE), A; cleaning of the first byte
RET; exit procedures
;-------------------------------------
; Merging the images (17 <= 21)
;-------------------------------------
; SCR_MRG
LD HL, (CONSTS); URL for an image taken from a cell
LD DE, # 4000; address display area
LP_SCRM: LD A, (DE); bytes screenshots
OR (HL); "leaked" to the byte image in memory
LD (DE), A; and placed back into the screen
INC HL; next byte images in the memory
INC DE; next byte of display area
LD A, D; checking for completion
CP # 58; display area
JR C, LP_SCRM; if not ended, then repeat
RET; exit procedures
;-------------------------------------
; Inverting Screen (12 <= 18)
;-------------------------------------
; SCR_INV
LD HL, # 57FF; last byte of the display area
LP_SCRI: LD A, (HL); took bytes screenshots
CPL; proinvertirovali it
LD (HL), A; and put back
DEC HL; move to the top of
BIT 6, H; if "handled" through the origin,
JR NZ, LP_SCRI; then STOP, otherwise twist cycle
RET; exit procedures
;-------------------------------------
; Inverting a vertical (20)
;-------------------------------------
; SINV_UD
LD HL, (CONSTS); taken from the cell address of the symbol
LD D, H; save this
LD E, L; address in DE
LD B, # 08; in the symbol - 8 bytes
LP_SIU1: LD A, (HL); take a one-byte characters
PUSH AF; and pushed on the stack
INC HL; the transition to the next byte characters
DJNZ LP_SIU1; repeat the cycle for the eight bytes
LD B, # 08; how many bytes will be read
LP_SIU2 POP AF; extract the bytes from the stack in reverse
LD (DE), A; Mr. procedure written in a symbol
INC DE; next byte characters
DJNZ LP_SIU2; twist cycle eight times
RET; exit procedures
;-------------------------------------
; Inverting symbol horizontally (17 <= 19)
;-------------------------------------
; SINV_LR
LD HL, (CONSTS); take out the cell address of the symbol
LD B, # 08; modify: 8 bytes
LP_SIL1: LD A, # 01; A set bit zero to 1
LP_SIL2: RR (HL); rotate byte characters left
RLA; and battery - to the left (via CF)
JR NC, LP_SIL2; until a zero bit will not appear in CF
LD (HL), A; write the modified byte
INC HL; next byte characters
DJNZ LP_SIL1; repeat cycle 8 times
RET; exit procedures
;-------------------------------------
; Rotation of the symbol in a clockwise direction (26 <= 42)
;-------------------------------------
; SROTATE
LD HL, (CONSTS); address the rotating character of the cell
LD B, # 08; 8 vertical columns in the symbol
LP_SRO1 PUSH HL; saved address on the
```

```
stack
LD A, # 80, included the seventh bit in the accumulator
LP_SRO2 RR (HL); rotate the bytes of the symbol to the right
RRA; and one bit of each byte
INC HL; gradually fill the battery
JR NC, LP_SRO2; until 7 incl. bits do not fall into the CF
POP HL; restore the address of the symbol
PUSH AF; satellite. column - character on the stack
DJNZ LP_SRO1; twist cycle by the number of columns
LD B, # 08; steel column lines - bytes
LP_SRO3 POP AF; pulls off bytes from the stack
LD (HL), A; and it is - has a new line character
INC HL; next byte characters
DJNZ LP_SRO3; repeat the number of lines (8 bytes)
RET; exit procedures
;---------------------------------------
; Change the attribute (16 <= 21)
;---------------------------------------
; ACHANGE
LD HL, (CONSTS); L - mask (AND), H - additive (OR)
LD DE, # 5AFF; the last byte of attributes
LP_ACHN: LD A, (DE); take the current value of the attribute
AND L; threw extra bits
OR H; added the necessary
LD (DE), A; and recorded in the old place
DEC DE; move to the top attributes
BIT 3, D; and not a schedule already?
JR NZ, LP_ACHN; if not, then twist cycle
RET; exit procedures
;---------------------------------------
; Change attribute (18 <= 22)
;---------------------------------------
; AREPLC
LD DE, (CONSTS); E - what to look for, D - than to replace
LD HL, # 5AFF; the last byte of attributes
LP_ARPL LD A, (HL); take a byte from the field attributes
CP E; Is not that looking for?
JR NZ, GO_ARPL; not jump change
LD (HL), D; yes, change to a new value
GO_ARPL DEC HL; move to the top region Tee attributes
BIT 3, H; attributes have not yet come to an end?
JR NZ, LP_ARPL; if not, then check the next
RET; exit procedures
;---------------------------------------
; Paint circuit (123 <= 263)
;123 = 88 +35 - together with the procedure POINT
;---------------------------------------
; PAINT
LD HL, (CONSTS); coordinates of the starting point
LD A, H; check the Y coordinate for output
CP # C0; beyond the screen:
RET NC; if Y> = 192, then the extra output
SBC A, A; because CF = 1, then SBC A, A gives A = # FF -
PUSH AF; this will be the end of the stack pointer
PUSH HL; remember the coordinates of first point
LP_PNT1: POP DE; take from the stack X, Y et seq. point
INC D; if Y = # FF, then the stack is exhausted,
RET Z; and then exit the procedure
DEC D; restores. Y value
CALL POINT; check point with coord-mi (E, D)
JR NZ, LP_PNT1; if enabled, the transition to the next.
EX AF, AF '; A' = 0, CF = 0 - aux. signs
LP_PNT2: LD A, E; took the X coordinate
OR A; if it is zero,
JR Z, GO_PNT1; then jump through the backward movement
DEC E; differently - reduce the X coordinate
CALL POINT; and check the previous point
JR Z, LP_PNT2; if "no obstacle", repeat
LP_PNT3: INC E; the transition to point to the right (X = X +1)
JR Z, LP_PNT1; if X> 255, ff. point from the stack
GO_PNT1: CALL POINT; check the following right point
JR NZ, LP_PNT1; if enabled, the trace. from the stack
LD A, (HL); if the point is not established,
OR C; then take a byte from the screen, turn on
LD (HL), A; the right bits and put back
LD A, D; check coordinate Y:
OR A; if it is zero,
```

```
JR Z, GO_PNT4; do not check the lying.
above the line
DEC D; the transition to the line above
(Y = Y-1)
CALL POINT; test points overlying
JR Z, GO_PNT2; if not included, the
transition
EX AF, AF '; have auxiliary flags
LD A, B; allowed to memorize a point in
the stack
JR GO_PNT3; the transition to the
continuation of
GO_PNT2 EX AF, AF '; have auxiliary
flags
INC A; if A> 0, it means a ban
DEC A; to save the new coordinates
JR NZ, GO_PNT3; point in the stack ->
jump
LD A, C; otherwise - do not store the
coordinates
PUSH DE; coordinates, but one shoves a
stack
GO_PNT3 EX AF, AF '; retained support
flags
INC D; return to the bottom line
GO_PNT4: LD A, D; check coordinate Y:
CP # BF; if - the latter (below do not
happen)
JR NC, LP_PNT3; the transition to the
next. point to
the right
INC D; otherwise - get down on the line
below
CALL POINT; check the underlying point
JR Z, GO_PNT5; if not included, the
transition
EX AF, AF '; have auxiliary flags
AND A; allowed to memorize a point in
the stack
JR GO_PNT6; the transition to the
continuation of
GO_PNT5: EX AF, AF '; have auxiliary
flags
JR C, GO_PNT6; if you can not save,
then go
SCF; forbid save point on the stack
PUSH DE; but one point on the stack
pushes
GO_PNT6: EX AF, AF '; retained support
flags
DEC D; return to the top of the line
JR LP_PNT3; the transition to the next
point on the
right
;--------------------------------------
; Check the status of the point and the
calculation of addresses in the screen
(35 <= 70)
;--------------------------------------
; POINT; if the point is disabled, ZF =
1 (Z), otherwise, ZF = 0 (NZ)
LD B, # 07; commonly used mask (# 07)
LD A, D; took the Y-coordinate
RRA; divided it into 8
SCF; and began to form
RRA; byte
RRA; address of the pixel
AND # 5F; in the screen (register H):
LD H, A;% 010yyyyy
XOR E; then form the
AND B; low byte
XOR E; address
RRCA; pixel
RRCA; in the screen
RRCA; (case L):
LD L, A;% yyyxxxxx
LD A, D; finish
XOR H; formation
AND B; High Byte
XOR H; address of the pixel
LD H, A; in the screen (register H)
LD A, E; begin forming
AND B; mask pixel in a byte
LD B, A; image (corresponding
LD A, # 80; bit included). Turn on the
seventh bit
JR Z, GO_PNT; if this is just what is
necessary,
LP_PNT RRCA; then jump through a shift
DJNZ LP_PNT; included bits to the right
GO_PNT: LD C, A; save mask inc. bit in
the reg. C
AND (HL); check the pixel in the screen
RET; exit procedures
;--------------------------------------
; Building templates (98 <= 196)
; 98 35 = 133 - together with the
procedure POINT
;--------------------------------------
; PFIGURE
LD DE, (CONSTS); coordinates of the
starting point
SLD HL, (23627), the start address of
the variables BASIC
LP_PFG1 LD A, (HL); the first byte of a
variable
INC HL; the transition to the next byte
LD BC, # 0012, the size of the loop
variable FOR ... NEXT
CP # E0; found a loop variable FOR ...
NEXT?
JR NC, GO_PFG2; if yes, go to the next.
Vac.
CP # 80; BASIC variables over?
RET Z; if so, exit procedures
LD C, # 05; long numbers. Vac. single
characters.
JR NC, GO_PFG3; array or a number. per.
```

```
with several. Sim.
CP # 60, the number of changes. single
characters. in a name?
JR NC, GO_PFG2; yes, go to the next
variable
CP "A"; the name of a character
variable (A $)
JR Z, GO_PFG4; cheers, still found and
and and and!
GO_PFG1: LD C, (HL); obtain
INC HL; the size of the study
LD B, (HL); variable
INC HL; in bytes, and
GO_PFG2: ADD HL, BC; adding to the
address
JR LP_PFG1; proceed to the next
variable
GO_PFG3: BIT 5, A; array variable?
JR Z, GO_PFG1; yes, jump over it
LP_PFG2: BIT 7, (HL); check at the end
it is the number. per.
INC HL; next byte name
JR NZ, GO_PFG1; name has ended, the
transition
JR LP_PFG2; continue to see the name
GO_PFG4: LD C, (HL); have long found
INC HL; string variable
LD B, (HL); with data on the pattern
LP_PFG3: INC HL; next character
template data
LD A, B; check: Do not run out
OR C; we all data on the pattern?
RET Z; if so (length = 0) then exit
DEC BC; reduced the length of
LD A, (HL); took the character template
data
CP "0" and not "put a point?
JR NZ, GO_PFG6; if not, go to the
continuation of
LD A, D; y-coordinate of current point
CP # C0; if the outside bottom edge of
the
JR NC, LP_PFG3; screen, the point does
not represent
PUSH HL; otherwise - keep some
PUSH BC; registers, not to spoil
CALL POINT; call validation points
LD A, (HL); on the calculated values
OR C; image point on the screen
LD (HL), A; using the fact that HL =
address
POP BC; and C register contains a mask
point
POP HL; restore the preserved-nye
registers
JR LP_PFG3; processing trace. wildcard
GO_PFG6: SUB "5" to shift the pen to
the left?
JR NZ, GO_PFG7; if not, leave it as is
DEC E; otherwise - to reduce the x-
coordinate
GO_PFG7: DEC A; move down?
JR NZ, GO_PFG8; no transition
INC D; yes, we increase the y-
coordinate
GO_PFG8 DEC A; direction "up"?
JR NZ, GO_PFG9; not jump
DEC D; yes, reduce the y-coordinate
GO_PFG9: DEC A; may need to move to the
right?
JR NZ, LP_PFG3; not, go to the next.
character of Chablis.
INC E; yes, we increase the x-
coordinate
JR LP_PFG3; and proceed to the next.
character of Chablis.
;-------------------------------------
; Increase the screen and up (174 <=
335)
; 174 35 = 209 - together with the
procedure POINT
;-------------------------------------
; PSCALER
LD HL, (CONSTS +4); magnitude of
increase in x and y
INC L; checking x-coordinate
DEC L; to zero
RET Z; if equal to 0, the error
(output)
INC H; checking the y-coordinate
DEC H; to zero
RET Z; if equal to 0, the error
(output)
LD HL, (CONSTS +6); new x-, y-
coordinate ("where")
LD A, # BF; maximum possible y-
coordinate
CP H; check the new y-coordinate
RET C; if not at the screen - Output
LD HL, (CONSTS); x1-, y1-coordinate
("source")
CP H; check y1 by entering the screen
RET C; if behind a screen, then exit
LD DE, (CONSTS +2); x2-, y2-coordinate
("source")
CP D; y2 is the screen?
RET C; if not, exit procedures
LD A, E; coordinate x2
CP L; compared with the coordinate x1
JR NC, GO_PSC1; if L <E, then
everything is fine
EX DE, HL; otherwise - have changed
their places
GO_PSC1: LD A, D; coordinate y2
CP H; compared with the coordinate y1
JR NC, GO_PSC2; if H 0, the CF flag
turned on
RR B; "twist" this bit in the register
```

```
B
DEC C; decrements the number of bits
JR NZ, GO_PSC3; if not zero, then jump
PUSH BC; else - throws a stack
INC SP; register B (only 1 byte)
LD C, # 08; and set the counter bits
GO_PSC3: LD A, E; current x-coordinate
DEC E; moving through the left
CP L; check on end line
JR NZ, LP_PSC2; twist cycle through
EX AF, AF '; recover the value of
LD E, A; x-coordinate of the
alternative A
LD A, D; current y-coordinate
DEC D;'re going to line up
CP H; it was the last line?
JR NZ, LP_PSC1; if not, then we turn
loop on lines
LD A, # 08; the number of bits in a
byte
SUB C; A = number of occupied bits in
reg. B
JR NZ, GO_PSC4; if not zero, then jump
LD A, C; A = C = 8 - number of bits in
a byte
DEC SP; remove from the stack the last
POP BC; abandoned there Bytes
GO_PSC4: LD C, A; how many bits of data
seq. byte
LD DE, (CONSTS +6); new x-, y-
coordinate ("where")
LP_PSC3: LD A, E; store x-coordinate of
the start
EX AF, AF '; line image A'
EXX; transition to alternative
registries
LD E, C; it will count for points x
LP_PSC4 EXX; back to Main. register set
EX AF, AF '; transition to Alt. flag.
Register
RLC B; flag CF - print / ne_vyv. point
EX AF, AF '; returned to normal flags
PUSH BC; store bytes of data and Acc.
bits
LD HL, (CONSTS +4); magnitude of
increase in x and y
LD B, H; kept the magnitude of
increases
LD C, L; in registers C and B (x and y)
PUSH DE; keep you coordinates (the loop
on lines)
LP_PSC5: PUSH DE; store coordinates you
(the cycle on points)
LP_PSC6: PUSH HL; save registers HL and
BC
PUSH BC; before calling the procedure
POINT
CALL POINT; calculation of addresses in
the screen and mask
LD A, C; mask points (bit included)
POP BC; BC restored from the stack
EX AF, AF '; check alternate CF flag
JR C, GO_PSC5; if it is enabled, then
jump
EX AF, AF '; keep this flag CF
CPL; invert the mask bits point
AND (HL); and use it to reset the pixel
JR GO_PSC6; the transition to the
continuation of
GO_PSC5: EX AF, AF '; CF flag again
make alternative nym
OR (HL); include pixel
GO_PSC6: LD (HL), A; write bytes
changed in the screen
POP HL; restore HL (sch. scale)
INC E; the transition to the lyrics.
point in the line of the screen
DEC L; decrements the scale of x
JR NZ, LP_PSC6; is not zero, continue
to cycle
LD L, C; recover the value of x-scale.
POP DE; Restore, it coordinates the
start line
INC D; the transition to the next line
in the screen
DEC H; decrements the scale of y
JR NZ, LP_PSC5; and steep until it
reaches 0
LD H, B; recover the value of y-scale.
POP DE; Restore, it coordinates the
start point
LD A, E; the transition to the
beginning of the next
ADD A, L; rectangle depicting
LD E, A; one point of the image (right)
POP BC; restore the byte to Dr. X and
the counter
DEC C; decrements of bits in a byte B
JR NZ, GO_PSC7; if there are bits, then
the transition
DEC SP; otherwise - are read from the
stack
POP BC; next byte of data in the
register B
LD C, # 08; set the counter bits
GO_PSC7: EXX; transition to alternative
registries
DEC E; decrements points in a row
JR NZ, LP_PSC4; if you still have a
point, then run around EXX; back to
Main. register set
EX AF, AF '; restore from an alternate
LD E, A; battery x-coordinate lines
LD A, D; the transition to the
beginning of the next straight.
ADD A, H; gon, depicts one
LD D, A; point of the image (down)
EXX; transition to alternative
```

```
registries
DEC B; reduce the row count sprite
EXX; back to Main. register set
JR NZ, LP_PSC3; cycle if the line did
not end
RET; exit procedures
;-------------------------------------
; end of 1940 New Best Routines
(graphic);
;-------------------------------------
```

Hex dump of the library

```
F270: 5A 11 FF 5A 01 02 E0 ED: F6
F278: B8 3A 00 5B 12 1D 20 FC: 02
F280: C9 12 11 00 40 62 6B 23: 8E
F288: 01 1F 00 ED B0 AF 13 12: 0B
F290: 7A FE 58 38 F0 C9 11 FF: 53
F298: 57 62 6B 2B 01 1F 00 ED: E6
F2A0: B8 AF 12 1B CB 72 20 F1: 74
F2A8: C9 D5 40 11 00 20 01 00: AA
F2B0: 7B 81 6F 7A C6 30 02 08: 87
F2B8: 67 ED B0 D1 7C FE 58 30: 81
F2C0: 12 14 7A E6 07 20 E5 7B: BF
F2C8: C6 20 5F 38 DF 7A D6 08: 6E
F2D0: 57 18 D9 AF 12 1C 20 FC: 03
F2D8: 1E E0 14 CB 5A 28 F4 C9: E6
F2E0: 11 FF 57 D5 20 01 00 7B: AA
F2E8: 91 6F 7A 30 02 D6 08 67: CB
F2F0: ED B8 D1 CB 74 12 28 7A: 4B
F2F8: 15 07 20 E6 E6 7B D6 20: 63
F300: 5F 38 E0 7A C6 18 08 57: 21
F308: DA AF 12 1D 20 12 FC 1E: FF
F310: 1F CB 15 72 20 F3 C9 21: 71
F318: FF 57 B7 CB 20 06 16 2B: 4A
F320: 10 FB CB 74 20 21 F4 C9: 5B
F328: 00 40 0E C0 B7 20 06 CB: D1
F330: 1E 23 10 FB 0D 20 F5 C9: 5A
F338: 11 00 40 62 6B 20 01 00: 6A
F340: 24 7C E6 20 07 09 7D 81: E7
F348: 6F 38 04 7C D6 08 67 E5: 8C
F350: ED B0 D1 7C FE 58 38 E3: 9E
F358: AF 12 1C 20 FC C9 11 FF: 1D
F360: 57 62 6B 20 01 00 7C 25: 39
F368: E6 20 07 09 7D 91 6F 38: 26
F370: 04 7C C6 08 67 E5 ED B8: A2
F378: D1 CB 74 20 E4 AF 12 1D: 5D
F380: 20 FC 12 C9 2A 00 5B 11: 00
F388: 00 40 1A B6 23 12 13 7A: 4D
F390: FE 58 38 F6 C9 21 FF 57: 47
F398: 7E 2F 77 2B CB 74 20 F8: 31
F3A0: C9 2A 00 5B 5D 54 06 08: A0
F3A8: 7E F5 10 23 FB 06 08 F1: 3B
F3B0: 13 12 10 FB C9 2A 00 5B: 21
F3B8: 06 08 3E 01 CB 1E 17 30: 28
F3C0: FB 77 10 23 F5 C9 2A 00: 40
F3C8: 5B 06 08 E5 3E 80 CB 1E: B0
F3D0: 1F 23 30 FA E1 F5 10 F3: 08
F3D8: 06 08 F1 77 10 23 FB C9: 38
F3E0: 2A 00 5B 11 FF 5A 1A A5: 81
F3E8: B4 12 1B CB 5A 20 F7 C9: C1
F3F0: ED 5B 00 5B 21 FF 5A 7E: 7E
F3F8: BB 20 01 72 2B CB 5C 20: AB
F400: F6 C9 2A 00 5B 7C FE C0: 72
F408: D0 9F F5 E5 D1 14 15 C8: 07
F410: CD 5A F4 20 F7 08 7B B7: 70
F418: 28 09 1D CD 5A F4 28 F6: 93
F420: 1C 28 E9 CD 5A F4 20 E4: 60
F428: 7E B1 77 7A B7 28 13 15: 43
F430: CD 5A F4 28 04 08 78 18: 03
F438: 07 08 3C 3D 20 02 79 D5: 24
F440: 08 14 7A FE BF 30 D9 14: A4
F448: CD 5A F4 28 04 08 A7 18: 4A
F450: 08 05 38 02 37 08 15 D5: B4
F458: 18 06 07 C6 7A 1F 37 1F: 26
F460: 1F E6 5F 67 AB A0 AB 0F: 24
F468: 0F 0F 6F 7A AC A0 AC 67: C2
F470: 7B A0 47 3E 80 28 03 0F: BE
F478: 10 FD 4F A6 C9 ED 5B 00: 7F
F480: 5B 2A 4B 5C 7E 23 01 12: 54
F488: 00 FE E0 30 13 FE 80 C8: E3
F490: 0E 05 30 0F FE 60 08 30: 6C
F498: FE 41 28 12 4E 23 46 23: DF
F4A0: 09 18 E1 CB 6F 28 F5 CB: B8
F4A8: 7E 23 20 F0 18 23 F9 4E: CF
F4B0: 46 23 78 B1 C8 0B 7E FE: 85
F4B8: 30 11 20 7A FE C0 30 F1: 66
F4C0: E5 C5 CD 5A F4 7E B1 77: 1F
F4C8: C1 E1 18 E5 D6 35 20 01: 87
F4D0: 1D 3D 3D 20 01 14 20 01: B1
F4D8: 15 3D 20 D5 1C 18 D2 2A: 43
F4E0: 04 5B 2C 2D C8 24 25 C8: 65
F4E8: 2A 06 5B 3E BF BC D8 2A: 22
F4F0: 00 5B BC D8 ED 5B 02 5B: 78
F4F8: BA D8 7B BD 01 30 EB 7A: 4C
F500: BC 02 30 54 67 7A 94 3C: E8
F508: D9 47 D9 7B 95 3C D9 4F: 6A
F510: D9 F5 33 0E 08 7B 08 E5: 84
F518: C5 CD 5A F4 C1 E1 C6 FF: 54
F520: CB 18 0D 04 20 C5 33 0E: 2F
F528: 08 7B 1D BD 20 E9 08 5F: EA
F530: 7A 15 BC 20 E0 3E 08 91: 47
F538: 20 03 79 3B C1 4F ED 5B: 5C
F540: 06 5B 7B 08 D9 59 D9 08: 2C
F548: CB 00 08 C5 2A 04 5B 44: A2
F550: 4D D5 D5 E5 C5 CD 5A F4: 01
F558: C1 79 08 38 05 08 2F A6: A9
F560: 18 02 08 B6 77 E1 1C 2D: CE
F568: 20 E9 69 D1 14 25 20 E2: DB
F570: 60 D1 7B 85 5F C1 0D 20: E3
F578: 04 3B C1 0E 08 D9 1D 20: 99
F580: C5 D9 08 5F 7A 84 57 D9: A8
F588: 05 D9 20 B6 C9 00 00 00: FA
```

Saving: SAVE "nbestg.c" CODE 62000,861

Deutsche Zusammenfassung

SerzhSoft stellt hier verbesserte „40 beste Maschinencode Routinen“ vor, welche optimiert wurden um weniger Speicherplatz zu verbrauchen. Da findet sich für jeden etwas, und für Spielecoder findet sich mit Sicherheit einiges an verwendbarem Code.
Die Listings stehen in Source-Form wie auch in Binärform zur Verfügung.
Ich muss mich für die Formatierung entschuldigen und für eventuelle Fehler. So wie es aussieht, wurde der Artikel eingescannt und maschinell übersetzt.
Ich habe zumindest versucht die gröbsten OCR-Fehler zu beseitigen, wie z.B. die Zahlenreihen im Hex-Listing als Datum interpretiert. Bei der Übersetzung habe ich dann aufgegeben (sie stammt nicht von mir), aber ich hoffe dass jedem Klar ist, dass z.B. mit Battery der Akkumulator gemeint ist.

Neuer Kontakt

von Norbert Opitz
Am 26.08.2012 früh um 9.00 Uhr, also während des Spectrum-Treffen in Wittenberg kontrolierte ich, ob ich neue E-Mails hatte. Darunter war eine leicht säuerliche von einem René Meyer, er beschwerte sich, daß wir um 19.00 Uhr nicht mehr an unseren Speccys zugange waren.
Nach einer erklärenden E-Mail und einem längeren Telefongespräch habe ich bei ihm einen Besuch gemacht.
Der René Meyer wohnt in Leipzig-Probstheida (Südost-Leipzig), Seidelstr. 2 a.
Der Besuch war am Montag, 3.September, ab 19.00 Uhr. Er wohnt mit Frau und drei kleinen Kindern in einem größeren Einfamilienhaus. Das Kellergeschoß hat er vollständig mit Computersachen belegt, und könnte es als Copmutermuseum bezeichnen.
Da sind in zwei Räumen 2x3 m Regale bis auf schmale Gänge und bis an die Decke hoch gefüllt mit Heimcomputern und Spielkonsolen von 1980er Jahren bis heute, darunter je ein ZX 81 und Gummi-Spectrum. In zwei weiteren Räumen 3x4 m stehen ringsrum an den Wänden Regale bis hoch an die Decke voll mit Computersoftware.
Nachdem er mir einiges von seinen Sachen gezeigt hatte, habe ich meinen Spectrum +2 mit dem MB 02 von Ingo Truppel vorführt und erklärt. Als Ergebnis meines Besuchs hat er uns zu der „Langen Nacht der Computer-Spiele“ eingeladen, und wir sollten unsere Computer mitbringen. Der Termin ist am Samstag dem 04. Mai 2013 ab 16.00 Uhr in Leipzig, Karl Liebknecht Str. 145 statt. Näheres findet man auf der Internetseite von René : www.schreibfabrik.de .

Retro Computer Match Teil 5

In Teil 1 wurde der TI99/4A vorgestellt, in Teil 2 der Atari XL, in Teil 3 war es Enterprise 64 und im Teil 4 hatten wir den Amstrad CPC.
Im Teil 5 tritt der Laser 210 gegen unseren ZX Spectrum an.

V-Tech stellt inzwischen Kinder-Lernlaptops her, doch damals hatten sie noch richtige Rechner im Programm, darunter auch Apple][Klone für die sie aus Lizenzgründen eigene kompatible Firmware programmiert haben, und so Apples Klagen (ja, die gab es damals auch schon) trotzten. Der Laser 110,

210 und 310 war aber eine eigene Architektur, die sich voneinander in der Speicherausstattung unterschieden haben, das Modell 310 bot außerdem noch eine Schreibmaschinientastatur.

	ZX Spectrum 48	Laser 210
Baujahr	1982	1983
CPU	Z80A 8 Bit	Z80A 8 Bit
Takt	3,5 MHz	4,7 MHz
RAM	48 KB	8 KB
VRAM	16 KB (geteilt)	2 KB
ROM	16 KB	16 KB?
Grafikmodi	1	1.
Textmodi	0	1
HwSprites	0	0
HwScroll	Nein	Nein
Auflösung	256x192	Bis zu 128 x 64
Farben	8 (15)	9 (4 bei „HiRes“)
Sound	1 Kanal	1Kanal
Soundchip	ULA	?
OS	BASIC	Laser BASIC
Medium	Kassette	Kassette
Speech	Softwaresample	Softwaresample
Tastatur	Folie und Gummi	Folie und Gummi

Die Vorteile gegenüber dem Spectrum waren: Schnellerer Prozessor (ich hatte eine Version mit 4,7 MHz (laut Anleitung) war üblich, anscheinend laut anderen Quellen waren es 3,25 MHz, allerdings stimmen die Daten nicht mit denen aus der Anleitung überein), und kleinerer Bildschirmspeicher (nur 2 KB!) wodurch schnellere Grafikeffekte möglich wurden. Natürlich war die geringe Auflösung dann sehr wohl hinderlich. Man muss es sich ungefähr so vorstellen wie wenn die Pixel nunmehr 2x3 ZX-Spectrum Pixel groß wären. Das ist eher Semi-Hires als Hires. Im Hires Modus waren somit 4 Farben möglich, während im Textmodus 9 Farben möglich waren.

Die Tastatur hat etwas bessere Folie als beim Spectrum, erinnert aber stark an diesen. Gummitasten hier und da... 45 Tasten waren es beim Laser 210, der Spectrum begann ab 40 Tasten.



Ausgegeben wurde das Bild auf einem Fernseher über RF-Antenneneingang, oder, was besser war, über Cinch Composite Video.

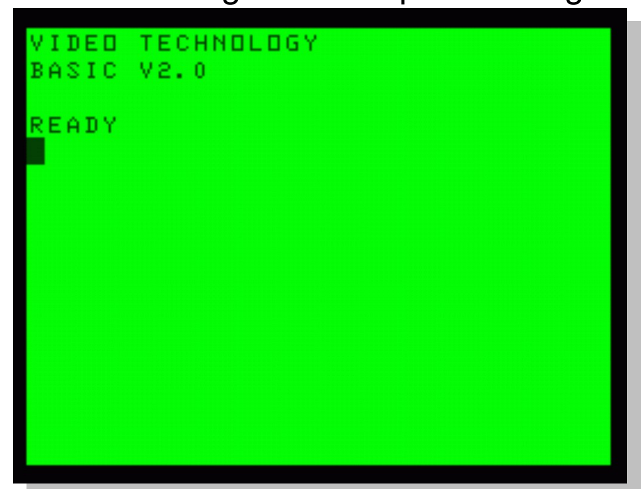
An Ports standen außerdem folgende zur Verfügung: Erweiterungsbus, Peripherie und Kassette. Als Zuberhör war ein Diskettenlaufwerk erhältlich.

Das in 16 Kb ROM eingebaute BASIC nannte sich Laser BASIC, hat aber nicht das geringste mit der Spiele-programmiersprache LASER BASIC für den Spectrum gemeinsam. Die neueste Version ist die 2.0, aber es sind Geräte im Umlauf mit früheren Versionen.

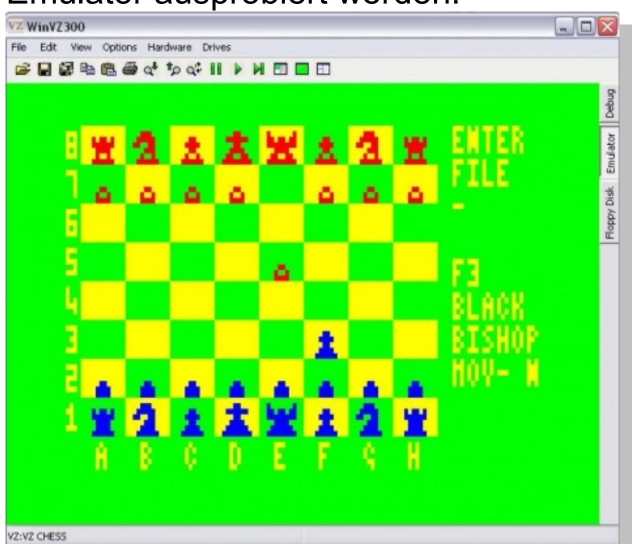
Die Befehle konnte man entweder eintippen, oder über ein Tastaturkürzel mittels Ctrl-Taste eingeben. Die Eingabe erfolgte im Textmodus mit 32x16 Zeichen.

Es gab einige Befehle die erst mit Extended BASIC eines damals 14-

jährigen Hobbyprogrammierers namens Russel Harrison freigeschaltet wurden. Das war das mächtigste Extended BASIC von einigen die erschienen sind, und es bot sogar einen Sprite-Manager.

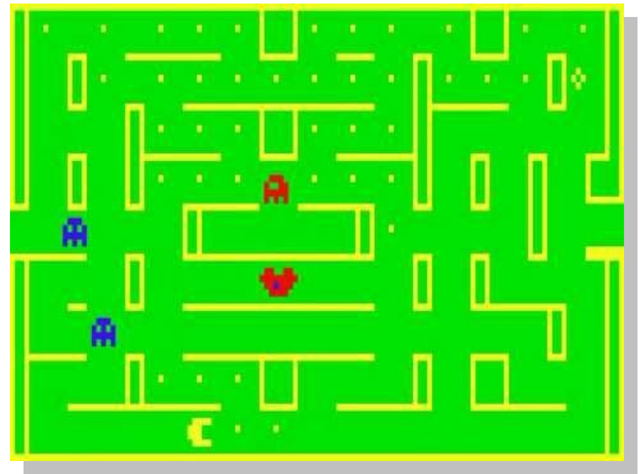


Populär war der Rechner nie und fristete immer nur ein Nischendasein, dennoch sind einige kommerzielle Programme entstanden die sich durchaus sehen lassen können.
Diese dürfen z.B. mit dem WinVZ300 Emulator ausprobiert werden:



http://www.t2e.pl/en/134/0/134/TryEmu/8306/Z80-WinVZ300-19122012#.UT5ul1fLsik

Während die User beim Spectrum mit hochauflösender Grafik und tausenden Programmen verwöhnt wurden, herrschte beim Laser eine flaute.
Natürlich gab es einige Hobbyisten die sich mit dem Gerät beschäftigten und immer noch beschäftigen, aber eine so große Szene wie am ZX Spectrum ist nicht entstanden.



Was Spiele betrifft, so wurden die damals populärsten Spielhallenspiele umgesetzt, wenn auch mit anderen Namen. Umsetzungen vom Spectrum kenne ich keine. Es gab Schach, Pac Man, City Defense, Moon Patrol und ähnliche.
Was neuere Spiele betrifft, so habe ich trotz intensivster Recherche kein einziges im Internet entdeckt.

Fazit: Hier siegt der Spectrum eindeutig. Auch wenn ich den Mut bewundere nach dem Spectrum auf den Markt mit einem Gerät schlechter und teurer als der Spectrum zu kommen, so musste ich dem Gerät eine klare Absage erteilen und verkaufte es wieder.
Das einzige Positive ist die schnelle Grafik, die sich vor dem Char-Modus nicht zu verstecken braucht.
Leider wurde dieses Feature selten ausgereizt.

LCD

WEISST DU MIT WEM SICH DEIN SPECTRUM HEUTE NACHT TRIFFT?

???



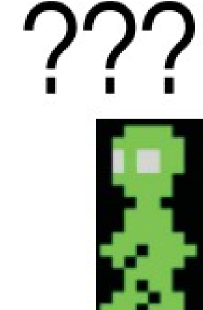
???

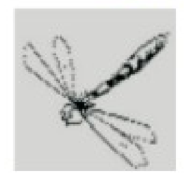


???



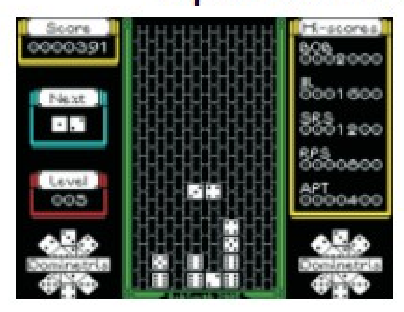


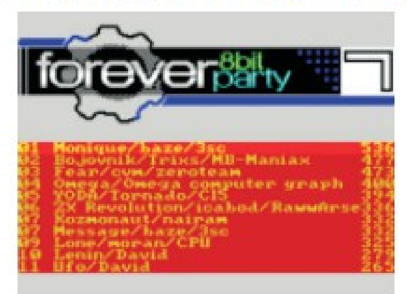




SCENE +

Spectrum-Entertainment starts here...







Einziges Spectrum Tape/Disk/Tap-Magazin
+D/Disciple, Opus, MB02, Tape oder .TAP Download
in Deutsch und Englisch

Erhältlich vom:

Scene+ Redakton, Mirko Seidel Neenstetter Str. 20, 89183 Breitingen
http://www.speccy-scene.de/ e-mail: ms-256-email@gmx.de

EIN MAGAZIN DES SPECTRUM-USER-CLUB, GERMANY